INPUT MSX

Publicación práctica para usuarios de MSX
Año 1 - Número 1
Abril 1986 - Precio 350 Ptas.

Contiene coleccionable

SECCION DE SOFTWARE

MSX ; hacia la compatibilidad

Envía mensajes secretos

Geometría fractal

TRINITRON

```
10 MSX
15 ---
20 PRINT "CANON"
25 X=DYNADATA
30 GOTO GOLDSTAR
35 IF JVC THEN
40 A$=MITSUBISHI
45 M=PANASONIC
50 GOSUB PHILIPS
55 LET PIONEER
60 IF SANYO THEN
65 SONY=A+1
70 C$=SPECTRAVIDEO
75 FOR TOSHIBA=1
80 GOTO YAMAHA
```


KNIGHT LORE
ULTIMATE
PLAY THE GAME


ULTIMATE
PLAY THE GAME



AÑO 1 NUMERO 1

OIRECTOR:
Alejandro Diges
OIRECTOR TECNICO:
Roberto Menéndez
COOROINAOOR EDITORIAL:
Francisco de Molina
DISEÑO GRAFICO:
Tomás López
COLABORADORES:
Antonio Taratiel, Luis R. Palencia, Francisco Tórtola, Benito Román, Esther de la Cal, Ernesto del Valle, Equipo Mollsoft.

INPUT MSX es una publicación juvenil de EDICIONES FORUM

GERENTE OIVISION DE REVISTAS:
Angel Sabat

PUBLICIDAD: Grupo Jota
Madrid: c/ Gral. Varela, 11
Telef. 270 47 02/03
Barcelona Avda de Sarriá, 11-13, 1.º
Teléf. 250 23 99

FOTOMECANICA: Ochoa, S. A.

COMPOSICION: EFCA, S. A.

IMPRESION: Edime, S. A
Depósito legal: M. 27 884-1985

SUSCRIPCIONES: EOISA,
López de Hoyos, 141. 28002 Madrid
Telef. (91) 415 97 12

REDACCION:
Alberto Alcocer, 46, 4.º
28016 Madrid. Telef. 250 10 00

DISTRIBUIDORA
R.B.A. PROMOTORA OE EOICIONES, S. A.
Travesera de Gracia, 56. Edificio Odiseus.
08006 Barcelona
El precio será el mismo para Canarias que para la Península y en él irá incluida la sobretasa aérea.

Se ha solicitado el control OJD

INPUT MSX es independiente y no está vinculada a MSX Research o sus distribuidores.

INPUT no mantiene correspondencia con sus lectores, si bien la recibe, no responsabilizándose de su pérdida o extravío. Las respuestas se canalizarán a través de las secciones adecuadas en estas páginas



SUMARIO

YA ESTAMOS AQUI

Salimos a la calle con muchas buenas intenciones, pero sobre todo con un objetivo: queremos ser vuestra revista preferida. Somos una publicación joven, hecha por gente joven y entusiasta, usuarios como vosotros. Además contamos con la experiencia que nos han proporcionado las revistas **INPUT Sinclair** e **INPUT Commodore,** hermanas nuestras.

Partimos de la idea de que una revista de usuario debe ser ante todo formativa, debe enseñar qué es lo que puede hacer un microordenador y debe explicar cómo hacerlo.

En esta línea, nuestros artículos tendrán como objetivo fundamental explicar las cosas de la forma más sencilla posible, con claridad y sin entrar en innecesarios tecnicismos.

Ni que decir tiene que además de formar, nuestra intención es también informar, por lo que podéis estar seguros de que os mantendremos al tanto de todas las novedades que se relacionen directa o indirectamente con vuestro **MSX.**

Os daremos a conocer los nuevos equipos, periféricos y accesorios que vayan apareciendo, y muy especialmente estaréis enterados de lo último en *software.* Mes tras mes comentaremos en detalle, explicando trucos y tácticas, los mejores programas que salgan al mercado.

Otro de los puntos fuertes de la revista queremos que lo constituya vuestra participación. Esperamos que nos enviéis montones de cartas. Para animaros, hemos pensado convocar unos cuantos concursos con sus correspondientes premios.

En las páginas centrales podréis encontrar un coleccionable con el que os enseñaremos BASIC a través de la programación de juegos. Empezaremos por lo más sencillo y poco a poco iremos complicando las cosas hasta llegar a juegos verdaderamente complejos.

Confiamos en que **INPUT MSX** llegue a ser la revista que siempre habíais deseado. Vamos a intentarlo y os invitamos, desde este mismo instante, a que nos ayudéis con vuestros comentarios y sugerencias.

EL BUZON DE INPUT

El próximo número de INPUT reservará un espacio para responder a vuestras cartas.

No pretendemos que sea un rincón únicamente destinado a la resolución de dudas y problemas. Por supuesto que intentaremos proporcionaros este tipo de ayuda, pero nuestra pretensión es tener algo más: un espacio para la comunicación donde nos contéis a todos lo que hacéis con el ordenador, proyectos, intenciones, cómo es vuestra instalación y las cien mil cosas que se os ocurran.

Dirigir las cartas a:

EL BUZON DE INPUT MSX
Alberto Alcocer, 46, 4.º B
28016 Madrid

MEJORAS Y APLICACIONES

Cualquier programa publicado en INPUT puede sugerir nuevas ideas. Te lanzamos una propuesta a modo de reto: mejora y/o encuentra aplicaciones a nuestros listados. No olvides que muchas veces salta una chispa que trae una solución genial.

Cada tercer mes, a partir de la publicación de un número, elegiremos la mejor aportación.

La decisión será inapelable, pero tener por seguro que será ecuánime y, sobre todo, resuelta por un grupo de expertos.

El ganador recibirá un premio consistente en una suscripción por un año a INPUT MSX.

HA LLEGADO LA NUEVA GENERACION

La segunda generación de ordenadores MSX no se ha hecho esperar demasiado. Son varias las firmas que han anunciado la comercialización de sus equipos MSX2 para los meses de Abril y Mayo. Concretamente Philips tiene previsto el lanzamiento de los nuevos equipos para los primeros días de Abril. El precio de venta al público de los mismos rondará las 130.000 Ptas. El equipo, que lleva incorporada una unidad de *diskettes* de 3.5 pulgadas, vendrá acompañado por un conjunto de tres paquetes *software* en *diskette* con: una base de datos, un tratamiento de textos en 80 columnas y un programa para diseno gráfico de amplias posibilidades. Philips comentó que no se esperan problemas de *software* para las nuevas máquinas, en primer lugar porque podrán utilizar todo el *software* de la primera generación, pero también porque las firmas involucradas en el estandar estan dispuestas a dar el soporte necesario con programas de todo tipo: lenguajes como LOGO, PASCAL, C, *software* profesional, *software* educativo y un estupendo *software* de juegos, con unos gráficos y una animación sorprendentes. Sólo queda por ver la respuesta de las firmas independientes de *software*.

SOFT EN SOFTCARD



A partir de Marzo, podremos encontrar programas para MSX en formato Softcard. Se trata de un soporte parecido a una tarjeta de crédito que incorpora un *chip* de ROM. Serma, empresa que inicia la producción de programas en este soporte, ha lanzado tres títulos: 1X2 y QH para los que esten pensando en hacerse millonarios y Le Mans 2, para los locos del volante. Proximamente aparecerá una versión de A View To a Kill, el programa sobre la película de James Bond. El precio de la tarjeta, con su adaptador, rondará las 6000 Ptas.

REPRESENTACION SPECTRAVIDEO

Se ha constituido, a partir del acuerdo firmado el pasado mes de Febrero con SVI International, la sociedad SVI España, dentro del grupo SVI Europe. Esta nueva empresa se ocupará de la comercialización en nuestro país, en exclusiva, de los productos Spectravideo, entre ellos el 738 X'Press y de los *joysticks* y otros productos Quickshot.

ORDENADORES EN LA ESCUELA

La Federación Española de Religiosos de la Enseñanza (FERE), que agrupa unas 2000 escuelas, en todo el territorio nacional, se ha decidido por la informática. Dicha federación ha llegado a un acuerdo con Philips, que aportará sus ordenadores MSX, tanto de la primera como de la segunda generación. Otros equipos seleccionados han sido los PCs de IBM y el modelo 520 ST de Atari.

MAS JUEGOS DE KONAMI



La empresa Serma firmó, el pasado mes de febrero, un acuerdo con Konami para la distribución, durante el presente año, de los programas de la firma japonesa. Los primeros títulos que se comercializarán a partir de este acuerdo serán: Hyper Sports III, con nuevas y divertidas pruebas deportivas y Yie Ar Kung-Fu II o Shaolin Road, para los aficionados a las artes marciales.

AVENTURAS EN MSX2

A partir de Mayo, Idealogic, lanzará al mercado la colección Telarium, para MSX2. Se trata de un conjunto de aventuras, que combinan musica, efectos especiales, gráficos y texto. Los programas son versiones de famosas novelas. Entre los 8 títulos que inician la serie figuran: "Cita con Roma" de Arthur C. Clarke, "Farenheit 451" de Ray Bradbury y títulos tan clásicos como "La Isla del Tesoro" y "El Mago de Oz". El formato será el de *diskettes* de 3.5 pulgadas.

LOS NUEVOS MITSUBISHI



Aparte del nuevo MSX2 de Mitsubishi, que se comercializará entre Abril y Mayo, Mabel, distribuidor en España, ha presentado novedades. Dos de ellas son el ML-FX1 y ML-FX2. El ML-FX2 incluye una ROM de 32K con 5 programas: Base de datos, hoja electrónica, tratamiento de textos, gráficos y comunicaciones.
El resto de las novedades lo componen la unidad de *diskettes* ML-03FD, de 720K, el cassette ML-10DR, el *joystick* ML-JY50 y un ratón para diseño gráfico, con las siglas ML-10MA.



EL ZOCO DE INPUT



Todo se compra y se vende. Los antiguos zocos fueron lugares destinados a todo tipo de transacciones. INPUT también tiene el suyo. Vuestras operaciones de compra, cambio o venta serán publicadas en esta sección, pero dos son las limitaciones que imponemos:
a) La propuesta tendrá que ver con la microinformática.
b) Nos reservamos el derecho de no publicar aquellos insertos de los que se sospeche un trasfondo lucrativo.
Ahora un ruego. Tratar de resumir al máximo el texto; escribir casi como un telegrama siendo claros y concisos.

Envía tu mensaje a:

INPUT MSX ZOCO
c/. Alberto Alcocer, 46
28016 MADRID

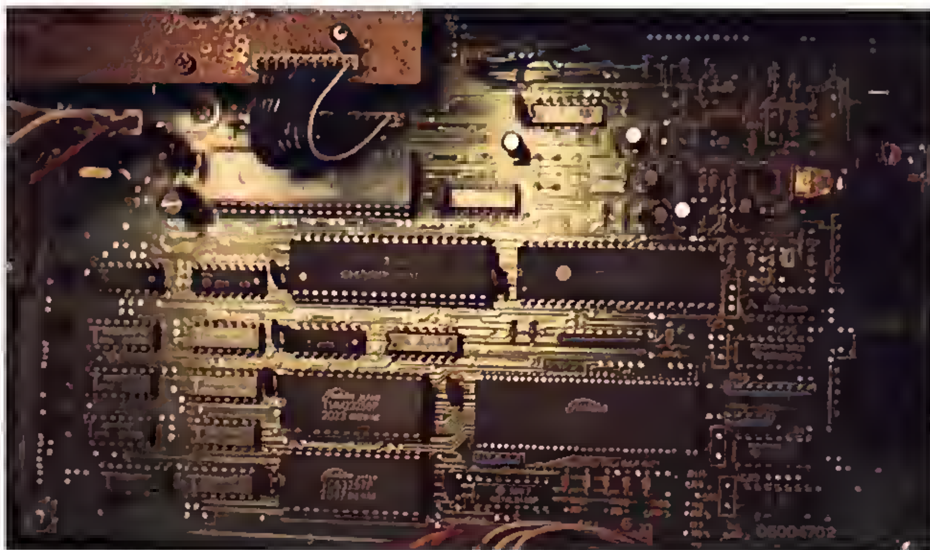
LOS SONIDOS DEL SILICIO

- INTERIOR DEL AY-3-8910
- LOS REGISTROS
- TERMINALES DEL CHIP
- PLAY Y SOUND
- ENVOLVENTES

Entre las características de los ordenadores MSX, **ocupa un lugar de importancia el tema del sonido. El estandar define al *chip*** AY-3-8910, **o cualquier otro equivalente, como generador de sonido de las máquinas** MSX.

En los primitivos ordenadores, el tema de la generación de sonidos apenas pasaba de ser algo anecdótico. La mayoría de estas máquinas a lo sumo incluían un pequeño zumbador, cuya única posibilidad era la de hacer «click» cada vez que recibía la orden, directamente de parte del microprocesador. Para llegar a generar algo que se pareciera a un sonido musical, aunque fuera un sonido simple, por ejemplo un tono puro, el programador se enfrentaba a la necesidad de escribir complicados programas en lenguaje máquina. La idea de estos programas era la de repetir a gran velocidad los «click» de forma que se obtuviera una señal periódica. Variando la velocidad de repetición se podían obtener diversos tonos, unos más graves y otros más agudos, con los que componer sencillas melodías. Además de ser un sistema poco flexible y de pocas posibilidades, había que dedicar mucho tiempo del microprocesador a tareas tan tontas como la de repetir, uno tras otro, la interminable serie de «clicks». El tiempo que se empleaba en esta tarea no podía utilizarse en otras, con lo





que si se quería escuchar un poco de música, había que olvidarse mientras tanto de los gráficos, o de cualquier otra tarea que necesitara del microprocesador.

Con el desarrollo de la tecnología de los circuitos integrados, y con el consiguiente abaratamiento de los *chips*, fué afianzándose la idea entre los constructores de ordenadores, de que lo mejor que se podía hacer era utilizar *chips* especializados, que se ocuparan de tareas específicas y que liberaran al microprocesador, dejándoselo al usuario. Es así como se llegó a los *chips* de vídeo, a los *chips* de control de periféricos, a los *chips* de comunicaciones, y también a los *chips* de sonido. Esta es la filosofía de las máquinas actuales y de las de un futuro próximo. Y esta es también la filosofía que ha presidido la definición del estandar **MSX**, tanto en su primera como en su segunda generación.

DENTRO DEL AY-3-8910

La labor musical en las máquinas **MSX**, (nos referimos a las de la primera generación), ha sido encomendada al **AY-3-8910**.

Este *chip* de **General Instruments**, ofrece unas interesantes características que le convierten en un estupendo generador musical. En su interior, una gran cantidad de dispositivos electrónicos se organizan en varios bloques cada uno con una misión muy específica. Los principales son: tres generadores de tono independientes, cada uno de los cuales proporciona sonido a uno de los canales exteriores, un generador de ruido blando, pensado sobre todo para la generación de efectos especiales, un generador de envolventes, que va a permitir dar diferentes formas y calidades a los sonidos producidos, un bloque de control de la amplitud o volumen de los sonidos, un sistema de mezcla para mezclar los tonos y el ruido en los distintos canales y por último, un bloque de conversión digital/analógica que va a ser el encargado de transformar las señales digitales que maneja internamente el *chip*, en señales analógicas, destinadas a ser reproducidas por un altavoz. Para controlar todos estos elementos, el *chip* dispone de 16 registros internos de 8 bits, a los que accede el microprocesador para dar las órdenes pertinentes e iniciar la generación de sonidos. El resto del *chip* está formado por circuitos para la comunicación con el microprocesador y por dos *ports* de entrada salida, que, aunque están dentro del *chip* de sonido, no intervienen en el proceso de generación musical, sino que se utilizan para la lectura de los *joysticks*.

Todos estos elementos vamos a poder controlarlos directamente desde nuestros programas. Para ello haremos uso de los 16 registros internos del *chip*, encargados de llevar un permanente control sobre el funcionamiento del resto de los elementos.

EL CHIP EN ACCION

Para que podamos llegar a escuchar, a través del altavoz de nuestro televisor, los estupendos sonidos que es capaz de producir nuestro *chip*, tienen que entrar en funcionamiento algunos o todos los elementos internos del mismo. Por ejemplo, si queremos oir la escala de notas musicales do mayor, que es la situada en el centro del teclado de un piano, podemos teclear lo siguiente:

```
10 PLAY "O4CDEFGABO5C"
```

Al pulsar RETURN escucharemos las notas DO, RE, MI, FA, SOL, LA, SI y un nuevo DO.

¿Qué es lo ocurrido dentro del *chip*?

Pues que se ha puesto en funcionamiento el generador de tonos del canal A y ha ido generando los tonos correspondientes a las distintas notas.

Para ello, los registros de control de frecuencia de este canal se han cargado sucesivamente con los valores correspondientes a la frecuencia de cada una de las notas. Una vez cargados en los registros los valores de cada nota, el generador de tonos se ha encargado de lo demás. Para ello ha puesto en funcionamiento un par de contadores internos y les ha obligado a contar hacia atrás, al ritmo del reloj del microprocesador, desde el valor que hay en los registros hasta cero, asf una y otra vez. Con esto, a la salida del generador de tono del canal A, ha aparecido una señal digital con una frecuencia correspondiente a la primera nota, al cabo del rato ha surgido la segunda nota, y así sucesivamente. El generador de tonos ha calculado la frecuencia de cada nota a partir de esta fórmula:

Frecuencia = Frecuencia Reloj / 16×(256×RA+RB)

Siendo:

Frecuencia Reloj - La frecuencia del reloj del ordenador, que para un **MSX** vale 1.78977 MHz.

RA - El contenido del registro de mayor peso de los que controlan la frecuencia del canal.

RB - El contenido del registro de menor peso.

La señal del generador de tono ha pasadò luego por el mezclador, pero como no queríamos hacer ninguna mezcla con ruido, que es lo que hace el mezclador, la señal ha seguido hacia adelante. Así ha llegado cada nota al conversor D/A y ha sufrido una transformación que la ha convertido en una nota analógica, con una amplitud y una envolvente determinadas. Con este aspecto, el conjunto de notas ha ido surgiendo del *chip*, a través de la patilla del canal A y han llegado hasta el altavoz, y de ahí hasta nuestro oído.

Si tecleamos ahora:

```
10 PLAY "04CD","EF","CA"
```

Podremos escuchar notas simultaneamente a través de los tres canales. En este caso hemos hecho funcionar simultaneamente los tres generadores de tono del *chip*. Si queremos mezclar ruido con la señal de uno de los canales, o simplemente escuchar ruido en el canal, tendremos, en primer lugar, que poner en funcionamiento el generador de ruido. Este funciona de una forma muy parecida a como lo hacen los generadores de tono. Un contador interno del generador empezará a contar desde el valor que hayamos introducido en el registro de periodo de ruido hasta llegar a cero, repitiéndose este proceso de forma continua. De esta forma se genera una señal de ruido con una frecuencia fundamental determinada (el ruido tiene componentes en una amplia gama de frecuencias). Después de generada, la señal de rudio llegará al mezclador y podremos sumarla a la señal de cualquiera de los canales. Para ello tendremos que activar ciertos bits en uno de los registros.

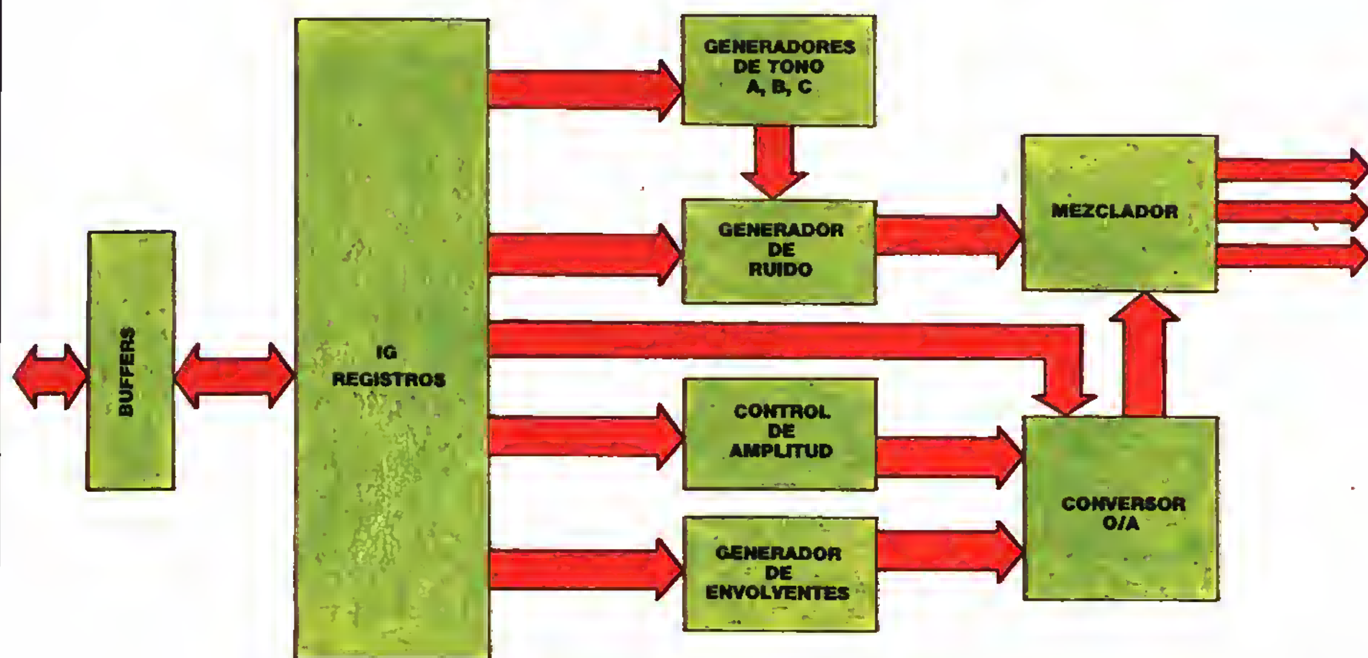
Vamos con otro ejemplo que nos ayudará a explicar el bloque de control de amplitud o volumen de cada uno de los canales.

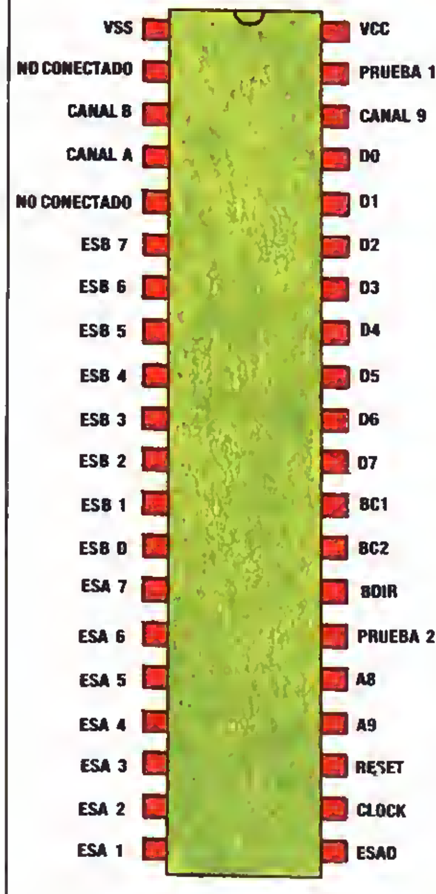
Prueba esto:

```
10 PLAY "04V1CDV15EFGV1AB"
```

Como habrás podido observar, se escuchan primero un DO y un RE de poco volumen, luego MI, FA y SOL mucho más fuertes, y por último un LA y un SI bastante flojos. Para conseguir este efecto hemos utilizado los parámetros V1 y V15 de la instrucción PLAY que quieren decir algo así como :pon el volumen a 1 y pon el volumen a 15. Dentro del *chip* lo que ha ocurrido es que el valor de uno de los registros, que controla la amplitud de la señal del canal A, se ha modificado entre unas notas y otras. Este registro actúa directamente sobre el bloque que controla la amplitud o el volumen en cada canal. Ha sido este bloque de control de amplitud el que ha actuado sobre la señal. Lo interesante de este control de amplitud es que puede actuar independientemente sobre cualquiera de los canales. De esta forma podremos, por ejemplo, escuchar una melodía a todo volumen por el canal A, mientras por el canal B se escucha un acompañamiento más suave. Además hemos visto que dentro de cada canal, se puede hacer variar el volumen entre dos notas consecutivas. Esta es una variación de volumen bastante rápida. Si no lo creéis, probad a hacer algo parecido con el mando de volumen de vuestro televisor mientras suena algo de Rock'n Roll.

Pero no queda ahí la cosa. El *chip* es capaz de variar el volumen mucho más deprisa, no sólo entre una nota y





la siguiente, sino mientras suena una sola nota. Es más, esta variación de volumen o de amplitud puede ser bastante compleja. De esta delicada labor se va a encargar el generador de envolventes.

SONIDOS Y ENVOLVENTES

Si escuchamos el DO que produce un piano y lo comparamos con el mismo DO de, por ejemplo, una flauta, enseguida notamos la diferencia. El que los dos sonidos sean el mismo DO se debe a que ambos son de la misma frecuencia. La diferencia entre ambos se debe a la forma de la onda. Son ondas de la misma frecuencia pero de forma distinta.

Si pensamos en términos de volumen, el DO del piano es una nota que ha crecido de volumen bruscamente (cuando hemos golpeado la tecla) y que luego, su volumen ha ido descendiendo poco a poco hasta desvanecerse en silencio. En cambio, el DO de la flauta, ha crecido de volumen igual o más deprisa que el del piano, pero luego, en lugar de desvanecerse lentamente, ha mantenido un volumen constante hasta el final de la nota, hasta que hemos dejado de soplar en la flauta. Esta variación rápida del volumen de un sonido es lo que se conoce como envolvente y es en gran parte responsable del timbre del sonido, es decir de su calidad, de lo que nos permite distinguir el sonido de un piano del de una flauta y estos del sonido de un acordeón. Es cierto que la envolvente no es lo único que define el timbre de los sonidos.

También interviene un factor denominado forma de onda. Así, el sonido de una flauta tiene una forma de onda más suave, más redondeada que el de un instrumento de metal, como por ejemplo la trompeta. La combinación de estos dos factores sí define totalmente el timbre del sonido. Si pudiéramos controlar ambos factores con la suficiente precisión, podríamos llegar a sintetizar cualquier sonido imaginable.

Con el *chip* **AY-3-8910** estamos algo más limitados. No podemos variar la forma de onda, pero sí tenemos un cierto control sobre la forma de la envolvente. Para ello vamos a tener que poner en funcionamiento el generador de envolventes. Este generador va a ser controlado mediante tres de los registros del *chip*. Dos de ellos van a determinar el periodo de la envolvente, el tercero su forma. Con este tercer registro vamos a poder generar hasta ocho formas distintas de envolvente. Algunas harán que el sonido se parezca más al de una flauta. Otras imitarán mejor el sonido de un piano. Para ver cómo suena cada una de las envolventes podemos teclear:

```
10 PLAY "SOM25004CDEFGABO5C"
20 PLAY "S4M25004CDEFGABO5C"
30 PLAY "S804CDEFGABO5C"
40 PLAY "S1104CDEFGABO5C"
50 PLAY "S1204CDEFGABO5C"
60 PLAY "S1304CDEFGABO5C"
70 PLAY "S1404CDEFGABO5C"
```

El programa toca la misma escala con envolventes diferentes. Se aprecia claramente cómo cambia el timbre de los sonidos. Las ocho formas de envolvente posibles las hemos representado en la figura. Cada forma viene definida por la combinación de los 4 bits menos significativos del registro de forma de envolvente. Cada uno de estos bits, que llevan los nombres de **CONT**, **ATT**, **ALT** y **HOLD**, tiene su propio cometido.

HOLD, que se puede traducir como sostenimiento o mantenimiento, determina si la envolvente mantiene o no, el valor final del primer ciclo. Si el bit está a cero, la envolvente se repetirá periódicamente, con la frecuencia que determinen los registros de periodo de la envolvente. Si el bit está a uno, la envolvente no se repetirá y el volumen del sonido se sostendrá, bien al máximo o a cero, según que el primer ciclo de la envolvente fuera ascendente o descendente.

ALT es el segundo de los bits de este registro. Se podría traducir como alternativamente. Si está a uno se sucederán alternativamente rampas ascendentes y descendentes que determinarán una envolvente triangular.

ATT es el bit de ataque. El ataque es el periodo inicial de la produceión del sonido, es decir, los primeros instantes, en los que el volumen pasa de cero (antes de que empiece el sonido) a un cierto valor distinto de cero (cuando ya se oye el sonido). El valor del bit **ATT** va a determinar si el ataque es ascendente, es decir, el volumen del sonido aumenta poco a poco, o bién si el ataque será descendente, esto es, el volumen será máximo al iniciarse el sonido para ir descendiendo poco a poco posteriormente.

CONT es el último bit de control. Va a determinar si la envolvente se repite periódicamente o nó, después del primer ciclo. Si este bit es cero, después del primer ciclo, el volumen del sonido caerá a cero.

SOUND Y LOS REGISTROS

Hemos comentado, aunque sin entrar en demasiadas profundidades, el

funcionamiento interno del *chip*. Para ello nos hemos sevido de algunos ejemplos que utilizaban la instrucción PLAY del lenguaje BASIC. Esta instrucción es realmente flexible y al mismo tiempo poderosa. Está pensada sobre todo para la creación, transcripción y ejecución de melodías y composiciones musicales.

PLAY va siempre seguido de una cadena de caracteres que es la que realmente contiene toda la información sobre la composición musical. Es como si PLAY le dijera al *chip* de sonido —!Prepárate, que lo que sigue es una cadena de caracteres con información de lo que tienes que hacer!—. Esta cadena de caracteres contendrá toda la información sobre la pieza musical, codificada en los siguientes caracteres: **C**,**D**,**E**,**F**,**G** ,**A** y **B** para las notas musicales DO,RE...(se trata de una notación anglosajona),**T** seguido de un valor entre 32 y 256 para la velocidad de ejecución, **V** seguido de un número entre 0 y 15 para determinar el volumen, **O** seguido de un valor entre 1 y 7 para la octava, **L** y **R** seguidos de los números 1,2,4,8,16,32 y 64 para la duración de notas y silencios respectivamente, **M** seguido de un valor entre 1 y 65535 para el periodo de repetición de la envolvente y, por último, S y un valor de 0 a 14 para determinar la forma de la envolvente.

Pero hay una forma más interesante de trabajar y, sobre todo, de experimentar con el *chip* de sonido **AY-3-8910**, que consiste en manejar directamente los dieciséis registros internos, de 8 bits, que controlan la actividad de todo el *chip*. Para ello el programador puede optar entre trabajar en lenguaje máquina o utilizar la instrucción SOUND del lenguaje BASIC.

SOUND es una instrucción muy sencilla, similar a la instrucción POKE, que permite modificar el contenido de los registros internos del *chip* de sonido. El formato de esta instrucción es:

SOUND número de registro, valor a introducir

Así por ejemplo:

```
10 SOUND 8,10
```

introduce el valor 10 en el registro número 8 del *chip* de sonido. Este registro controla el volumen de la señal del canal A, por lo que la anterior instrucción fija el volumen del canal A en 10 (el volumen puede variar entre 0 y 15, siendo cero el mínimo y quince el máximo).

SOUND nos va a permitir trabajar directamente con el *chip* de sonido desde el lenguaje BASIC. Es una instrucción mucho menos potente que PLAY, es decir, para hacer lo mismo que con una sola instrucción PLAY, vamos a necesitar de varias instrucciones SOUND. Pero esta falta de potencia queda compensada por una mayor flexibilidad. Por ejemplo con PLAY no vamos a poder hacer uso del generador de ruido, elemento importante a la hora de crear efectos especiales, en cambio si vamos a poder utilizarlo a través de la instrucción SOUND.

Vamos a completar la breve mención que, sobre el cometido de algunos de los registros, hicimos en el apartado dedicado al funcionamiento interno del *chip*. El conocimiento y el control de cada uno de ellos es lo que nos va a permitir explotar al máximo las posibilidades del *chip*.

De los dieciséis registros internos, numerados de 0 a 15, los 14 primeros están dedicados al sonido, mientras que los dos últimos controlan los *ports* de entrada y salida del *chip*, dedicados a la lectura de los *joysticks*. Pasemos a describir brevemente cada uno de estos catorce registros.

0 y 1: Estos dos primeros registros tiene la misión de controlar la frecuencia del generador de tonos del canal A. Según el valor que introduzcamos en ellos, se generará un tono más grave o más agudo, de mayor o menor frecuencia. Los ocho bits del registro de menor peso, el cero, se ocuparán del ajuste fino de la frecuencia. Los cuatro bits del registro uno harán lo mismo con el ajuste grueso. El contenido de ambos determinará la frecuencia del tono según la fórmula que vimos al hablar del funcionamiento interno del chip.

2 y 3: Igual que los dos anteriores, pero en este caso controlando el canal B.

4 y 5: Lo mismo pero con el canal C.

6: Este es el registro que controla el periodo de la señal de ruido. Sólo sus primeros cinco bits influyen en esta frecuencia.

7: Se trata del registro de activa-

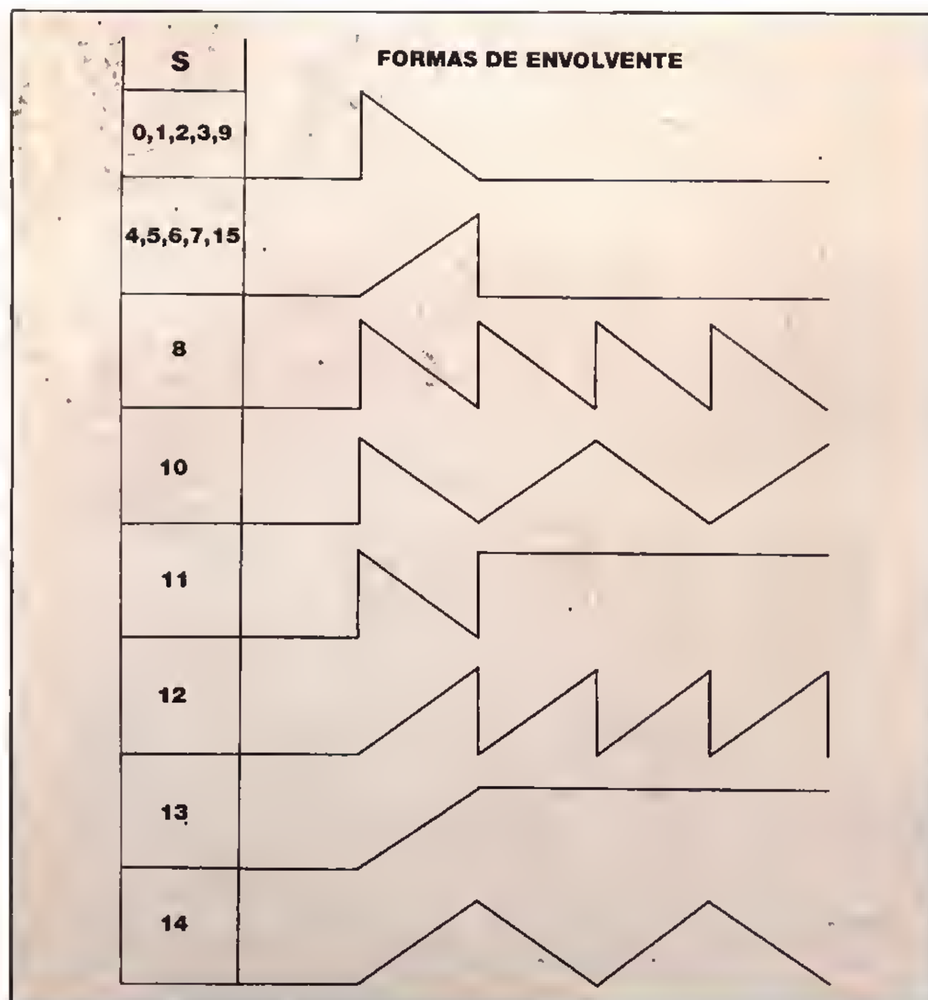
REGISTROS DEL AY-3-8910

REGISTRO	BIT	BT	B6	B5	B4	B3	B2	B1	B0
0	PERIODO DE LA SEÑAL DEL CANAL A	8 BITS DE AJUSTE FINO							
1						4 BITS DE AJUSTE GRUESO			
2	PERIODO DE LA SEÑAL DEL CANAL B	8 BITS DE AJUSTE FINO							
3						4 BITS DE AJUSTE GRUESO			
4	PERIODO DE LA SEÑAL DEL CANAL C	8 BITS DE AJUSTE FINO							
5						4 BITS DE AJUSTE GRUESO			
6	PERIODO DEL RUIDO				5 BITS DE AJUSTE				
7	ACTIVACION	E/S		RUIDO			TONO		
		IOB	IOA	C	B	A	C	B	A
8	AMPLITUD CANAL A				M	V3	V2	V1	V0
9	AMPLITUD CANAL B				M	V3	V2	V1	V0
10	AMPLITUD CANAL C				M	V3	V2	V1	V0
11	PERIODO DE LA ENVOLVENTE	8 BITS DE AJUSTE FINO							
12		8 BITS DE AJUSTE GRUESO							
13	FORMA DE ENVOLVENTE					CONT	ATT	ALT	HOLD
14	E/S PORT A	8 BITS DE E/S PARALELO							
15	E/S PORT B	8 BITS DE E/S PARALELO							

ción, que se va a encargar de controlar la mezcla de señales en los distintos canales. Los tres bits de menor peso, van a determinar si se asigna o nó cada uno de los generadores de tono a su canal correspondiente. Si el generador de tono se asigna al canal, poniendo el bit correspondiente a uno, entonces se escuchará el tono. Si por el contrario, no se asigna, no se escuchará el tono por el canal, aunque si podrá escucharse ruido si se ha activado uno de los tres bits que vamos a ver a continuación. Estos, que son los tres bits siguientes, van a determinar si se asigna el único generador de ruido a alguno, ninguno o todos los canales. Si se asigna ruido, a un canal, poniendo a uno el correspondiente bit, se escuchará la señal de ruido por el canal. Si además dicho canal tenía asignado su generador de tono, lo que se escuchará por el canal será la mezcla, o mejor dicho la suma, de las señales de tono y ruido. Esta mezcla permitirá obtener resultados sumamente interesantes. Los dos bits de mayor peso de este registro, se encargan del control de los *ports* de entrada y salida.

8,9 y 10: Estos tres registros van a controlar la amplitud o volumen de la señal de cada uno de los tres canales. En los cuatro bits de menor peso de cada uno de estos registros se puede introducir un valor entre cero y quince. Este valor será el volumen de la señal en el canal y consiguientemente en el altavoz. En cuanto al quinto bit, que es el último de los que se utilizan en estos registros, determinará si sobre cada uno de los canales actúa el generador de envolventes o sólo el control de volumen mencionado.

11 y 12: Estos dos registros determinarán la frecuencia de repetición de la envolvente, en el caso de que sea periódica.



13: Este es el registro de forma de envolvente. Sus cuatro primeros bits reciben los nombres de **HOLD, ALT, ATT** y **CONT** y, como ya hemos visto al hablar de la envolvente, nos van a permitir fijar la forma de la misma, escogiendo entre ocho formas diferentes.

LOS TERMINALES DEL CHIP

En la figura hemos representado el *chip*, en una vista superior, y hemos dado nombre a cada uno de los 40 terminales o patillas del mismo. Cada uno de estos terminales tiene una misión muy concreta que vamos a explicar. Empecemos por la alimentación. El *chip* se alimenta a una tensión de cinco voltios, que hay que aplicar entre los terminales Vss (masa o cero voltios) y Vcc (5 voltios). Los terminales señalados como **ESA0...ESA7** y **ESB0...ESB7**, corresponden a los dos *ports* de entrada/salida del *chip*. **D0...D7** son los terminales del bus de datos, aunque también se utilizan para transmitir la dirección del registro interno al que van destinados los datos. El que su contenido, en un momento dado, sea una dirección o un dato, dependerá del estado de las líneas o terminales **BC1**, **BC2** y **BDIR**. Los terminales denominados CANAL A, CANAL B y CANAL C, son las salidas analógicas de los tres canales de sonido del *chip*. Estas señales se mezclan y se envían al altavoz. Los terminales de **PRUEBA** no influyen en el funcionamiento del *chip*, sólo los utiliza el fabricante para comprobar el funcionamiento del circuito. El terminal marcado **CLOCK**, es la entrada de la señal de reloj del microprocesador. Esta señal es la que marca el ritmo de funcionamiento de todo el circuito. El terminal de **RESET** se utiliza en la iniciación del *chip*. Cuando se pone a cero, se borra el contenido de todos los registros.

Por último, los terminales **A8** y **A9** sirven para seleccionar el *chip*. Sólo cuando estos terminales estén a uno y a cero respectivamente, el *chip* estará seleccionado y podrá recibir datos y comunicarse con el microprocesador.

PENSANDO EN LA UNIDAD DE DISCOS

- ALMACENAR INFORMACION
- LAS UNIDADES DE DISCO
- TIPOS DE FICHEROS
- COMANDOS DEL MSX DISK BASIC

Los sistemas de almacenar información son elementos de gran importancia a la hora de trabajar con un ordenador.

Por una parte, todo ordenador tiene una memoria RAM, en la que se pueden guardar todo tipo de datos. El problema es que esta memoria se borra cuando se desconecta el aparato, con lo que toda la información almacenada desaparece y no podremos volver a recuperarla. Luego a la hora de almacenar nuestra información de forma permanente, la memoria RAM no sirve. Los ordenadores incluyen otro tipo de memoria de característias distintas, se trata de la memoria ROM (*Read Only Memory*) o memoria sólo de lectura. Esta memoria ROM es permanente, es decir, no pierde la información al quedarse sin corriente, por lo que en principio podría pensarse que es la memoria que necesitamos para guardar nuestra información. Sin embargo no es así. La memoria ROM no se borra, efectivamente, pero es porque contiene información vital para el ordenador, parte de lo que se conoce como sistema operativo, es decir un conjunto de programas y rutinas necesarias para que funcione el ordenador. El caso es que nosotros no podemos modificar esta información, no podemos por ejemplo guardar un listín de direcciones en la memoria ROM, ni tampoco un programa BASIC o una rutina en lenguaje máquina. La memoria ROM tampoco es para que guardemos nuestros datos.

Hay otros medios. Son los sistemas de almacenamiento secundario (el almacenamiento primario es el que se hace en memoria RAM) también conocidos como sistemas de almacenamiento masivo (pues permiten guardar mucha información). Entre estos sistemas están, por ejemplo, las memorias de burbujas y los discos duros, con capacidades variables entre unos 10 y varios cientos y hasta miles de Megabytes (un Megabyte es algo más de un millón de bytes). Y luego naturalmente están los conocidos sistemas de grabación de información en *cassette* (con capacidades variahles entre varios cientos de Kilobytes y algún Megabyte) y los *diskettes*, cuyas capacidades oscilan entre los 200 Kilobytes y un Megabyte aproximadamente. Estos dos últimos serán los sistemas que utilizaremos preferentemente para almacenar nuestros datos, programas e información de cualquier tipo.

UNIDADES DE DISCOS Y DISKETTES

Hasta ahora, el medio más utilizado para almacenar la información de un ordenador doméstico ha sido siempre el *cassette*, fundamentalmente por una cuestión de precio. Sin embargo, cada vez resultan más populares y más asequibles las unidades de discos.

Las ventajas del *diskette* se centran en varios aspectos muy concretos. Uno de ellos es la forma de acceso a la información, muy relacionada con la velocidad de transferencia de la misma. En este aspecto las unidades de *diskette* son esencialmente más rápidas que el *cassette* y ello no sólo por la propia velocidad de transferencia, que para el *cassette* es de unos 600 a 2400 bits por segundo mientras que para el *diskette* puede alcanzar valores del orden de los 250000 bits por segundo, sino por el hecho de que en el *diskette* se puede acceder a cualquier información directamente. Por el contrario, en el *cassette* hay que acceder secuencialmente, pasando primero por toda la información que hay antes de la que buscamos.

A este aspecto de la velocidad hay que añadir el de las opciones de trabajo. Con un *cassette* estamos bastante limitados. Sin embargo con un *diskette* dispondremos normalmente de una especie de sistema operativo para el manejo del *diskette*, es decir, de un conjunto de programas y rutinas que nos ayudarán a la hora de manejar la información. Este conjunto de rutinas



son a veces la base de todo un sistema operativo. Esto ha ocurrido en máquinas profesionales, por ejemplo con el sistema operativo **CP/M** o con el **MS-DOS**, sistema operativo de **Microsoft**. Por cierto, **DOS** significa Sistema Operativo en Disco (*Disk Operating System*). Aunque el estandar **MSX** no incluye unidades de *diskette* casi todos los fabricantes han lanzado al mercado unidades de este tipo. Dentro de los distintos formatos existentes, se ha escogido el de *diskettes* de tres y media pulgadas. El sistema consiste normalmente en una unidad de *diskettes* y en un *interface*, que se conecta al conector de cartuchos. Este *interface* lleva incorporado un *chip* de ROM con el sistema operativo para el manejo de los *diskettes*. A este sistema operativo se le conoce como **MSX DISK BASIC**, ya que proporciona comandos para utilizar desde el lenguaje BASIC. Por supuesto, existe una estandarización tanto en el sistema operativo de disco, como en los conectores e *interface*. De esta forma, se puede conectar una unidad de *diskettes* de cualquier fabricante, en el ordenador de cualquier otro fabricante, con la seguirdad de que no vamos a tener problemas de compatibilidad. Es más, también hay compatibilidad en cuanto a los formatos de grabación de datos en los *diskettes*, lo que quiere decir que si un amigo nuestro nos presta un *diskette* con programas grabados en su ordenador, podemos tener la seguridad de que funcionarán perfectamente en el nuestro. Esta es una de las ventajas de ser usuario de un **MSX**.

FICHEROS EN DISKETTE

En nuestro *diskette* vamos a poder almacenar cualquier tipo de información que se nos ocurra. Por ejemplo



programas en BASIC o en cualquier otro lenguaje como LOGO, PASCAL o lenguaje máquina. Pero no sólo programas, sino también, por ejemplo, pantallas gráficas, textos escritos, los nombres, direcciones y teléfonos de un listín telefónico, en fin, cuanta información se nos ocurra y queramos guardar. Toda esta información va a aparecer en el *diskette* agrupada en lo que se conoce como ficheros. Un fichero va a ser el conjunto de información que vamos a manejar de una vez, mediante los comandos que nos proporciona el **MSX DISK BASIC**.

Cuatro son los tipos de fichero que podemos manejar directamente desde el **MSX DISK BASIC**. En primer lugar están los programas BASIC. Es el tipo de fichero que se obtiene cuando guardamos un programa BASIC con el comando:

SAVE «NOMBRE DEL FICHERO»



En un fichero de este tipo, lo que se almacena no son los caracteres que podemos leer en la pantalla, y que constituyen nuestro programa, sino una versión comprimida del mismo. De esta forma se ahorra espacio de almacenamiento. Lo que se almacena son valores hexadecimales, conocidos como *tokens*, que representan a cada una de las instrucciones del programa.

Otro tipo de fichero es el que se obtiene cuando guardamos en el *diskette* un trozo de memoria, mediante el comando:

BSAVE «NOMBRE DEL FICHERO», dirección inicial, dirección final

En este caso se suele hablar de ficheros en lenguaje máquina o en código máquina. Lo que guardamos en el fichero son los valores binarios correspondientes a un trozo de memoria. Estos valores binarios, pueden ser un programa en lenguaje máquina, una imagen de pantalla, etc. Los dos tipos restantes de ficheros están pensados para que el usuario guarde datos de caracter general. Son los ficheros secuenciales y los ficheros de acceso directo.

FICHEROS SECUENCIALES Y DE ACCESO DIRECTO

En un fichero secuencial, los datos se almacenan uno tras otro a medida que llegan al fichero. Cada dato ocupará una cierta cantidad de bytes en el *diskette*, según su longitud. Por ejemplo, si queremos almacenar en *diskette* un pequeño diccionario, podremos utilizar para ello un fichero secuencial en el que almacenaremos las palabras de la siguiente forma:

PALABRA 1, SIGNIFICADO 1
PALABRA 2, SIGNIFICADO 2
PALABRA 3, SIGNIFICADO 3
ETC.

Al crear el fichero, primero se almacenará la palabra 1, a continuación su significado, luego la palabra 2, seguida del significado 2, etc. Lo importante de esta forma de almacenar la inforamción es que se aprovecha muy bien el espacio disponible en el *diskette*, pues entre una palabra y la siguiente no sc deja espacio libre. Pero hay un pequeño problema, y es que no sabemos de antemano en que posición encontraremos por ejemplo la palabra 10, por lo que si queremos leer dicha palabra, tendremos que leer antes las 9 palabras y los 9 significados que la preceden. Esto nos va a hacer peder algo de tiempo. Muy poco en el caso de 10 palabras, pero imaginemos lo que ocurriría con un diccionario de 10.000 palabras, por ejemplo, en el que tuviéramos que leer repetidamente palabras situadas al final del diccionario. Las demoras de tiempo se irían acumulando y al final nos encontraríamos con bastante tiempo perdido.

Para solventar este tipo de problemas tenemos a nuestra disposición los ficheros de acceso directo. En ellos, la información se guarda en bloques de longitud fija que hay que prefijar de antemano. Por ejemplo, podríamos crear un diccionario en el que cada palabra ocupara 20 bytes y su significado pongamos 80 bytes. De esta forma habríamos dividido el fichero en bloque de 100 bytes. A esto bloques se les suele dar el nombre de registros. En el primer bloque estarían la palabra 1 y el significado 1, en el segundo la palabra 2 y su significado y así sucesivamente. Supongamos ahora que hemos guardado de esta forma 10.000 palabras y que queremos buscar la palabra 7.000. No nos hará falta empezar a leer palabra tras palabra, como en el caso de un fichero secuencial, hasta llegar a la número 7.000. Como sabemos que cada palabra ocupa 100 bytes, podemos deducir que la palabra 2 empezará en el byte 100, la 10 en el byte 900 y la 7.000 en el byte 699.900. Podremos ir directamente a dicho byte y empezar a leer. De esta forma ahorraremos un montón de tiempo. Claro que no todo son ventajas en un fichero de acceso directo. Aunque ahorraremos tiempo a la hora de buscar la información, también desperdiciamos espacio. Por ejemplo si en uno de los bloques de 100 bytes, la palabra y su significado ocupan en total 50 bytes, hemos desperdiciado los otros 50 bytes, que no contendrán información. Esto no ocurría en el caso de un fichero secuencial, en el que las palabras y significados iban unos a continuación de otros.

COMANDOS PARA MANEJAR FICHEROS

Una de las ventajas de la utilización de *diskettes* es, como ya hemos mencionado, la de disponer de los comandos del sistema operativo para el manejo de *diskettes*. Este conjunto de comandos son la parte esencial del **MSX DISK BASIC**. Para que podaís apreciar todas las posibilidades que ofrecen estos comandos, los hemos agrupado, según su función, en varias categorías que vamos a comentar. No pretendemos que esto sea un manual de comandos, sino sólo dar una idea de lo que se va a encontrar el usuario que adquiera una unidad de *diskettes*. En primer lugar nos encontramos con los comandos más generales, que nos van a servir para manejar *diskettes* y ficheros, cualquier tipo de fichero, sin importar cual sea su formato. Los comandos **CALL FORMAT**, **FILES**, **COPY** y **KILL**, son de este tipo.

FORMAT sirve para establecer el formato de grabación en los *diskettes* vírgenes. Se definen 80 pistas concéntricas, cada una con 9 sectores, en cada uno de los cuales se almacenan 512 bytes, lo que nos da un espacio para almacenar datos del orden de los 360 Kilobytes.

FILES proporciona un listado de todos los ficheros que hay en cada *diskette*.

COPY es un comando muy interesante, presente en todos los sistemas operativos de disco calificados de serios, y pensando sobre todo para el trabajo con dos unidades de *diskettes* simultáneamente. Se utiliza fundamentalmente para transferir ficheros de un *diskette* a otro.

KILL (que en inglés significa matar) se utiliza para borrar todo tipo de ficheros de los *diskettes*.

Como en otros sistemas operativos de disco, en **MSX DISK BASIC** se pueden combinar todos estos comandos con los símbolos * e ?, de modo que el comando actúe sobre un conjunto de ficheros en lugar de sobre

uno solo. Por ejemplo el comando:

FILES «A:NUM*»

nos dará un listado de todos los ficheros de la unidad de *diskettes* A que empiecen por NUM. La utilización de estos caracteres (conocidos como *wild cards*) proporciona bastante flexibilidad cuando se manejan muchos ficheros. También en esta línea, nos encontramos con comandos como **DSKF** que nos dice cuanto espacio disponible nos queda en el *diskette*, **LOF** que nos da la longitud en bytes de cualquier fichero, **MERGE**, comando utilísimo que nos permite mezclar o unir ficheros, y **NAME** para cuando queramos cambiar el nombre de un fichero.

COMANDOS MAS ESPECIFICOS

Aparte de estos comandos de caracter general, tenemos a nuestra disposición otros, específicos para cada uno de los tipos de ficheros que podemos manejar. Así...

BSAVE, **BLOAD**, **DSKO** y **DSKI$** nos permitirán hacer maravillas con el código máquina o con bloques de memoria de cualquier tipo, por ejemplo imagenes de la memoria de vídeo.

SAVE y **LOAD** serán los que utilicemos para nuestros programas BASIC.

OPEN, **CLOSE**, **INPUT#**, **LINE INPUT#** y **PRINT#**, nos servirán para trabajar cómodamente con los ficheros secuenciales. Y por último...

GET#, **PUT#**, **LSET**, **FIELD**, **CVI**, **CVS**, **CVD**, **MKI$**, **MKS$** y **MKD$** serán comandos que simplificarán enormemente la tarea de trabajar con ficheros de acceso directo.

Toda esta maraña de comandos o palabras clave no tiene que asustarnos. Están ahí para hacernos las cosas más fáciles, para servirnos como herramientas con las que trabajar eficiente y cómodamente. Son los comandos de un sistema operativo de disco, el **MSX DISK BASIC**, con el que podremos hacer muchas cosas interesantes. En próximos números os iremos enseñando cómo.

SIMULACION: ALIMENTO PARA LA MENTE

- SIMULACION DEL CLIMA
- VARIABLES NORMALMENTE DISTRIBUIDAS
- CONDICIONES COMERCIALES
- TENEDURIA DE LIBROS

Anímate a escribir programas utilizando un método de creación de modelos eficaz, comprobando cómo pueden aplicarse provechosamente los principios del mundo de los negocios a la gestión de un puesto callejero de comidas.



La creación de modelos es un aspecto de gran importancia de la ciencia matemática, sobre el que se han vertido rios de tinta. Aún hoy en día, las teorías matemáticas no suelen alcanzar el favor del público, especialmente si no pasan del terreno de las hipótesis y no se traducen en algo de aplicación práctica. Pero también en este campo, el advenimiento de los microordenadores ha modificado la tradicional actitud pasiva frente a los problemas matemáticos. De hecho, la potencia de cálculo, presente incluso en el más modesto de los microordenadores, ha eliminado uno de los principales obstáculos psicológicos en la aplicación de las leyes y teoremas matemáticos: lo tedioso de unos cálculos manuales, repetitivos, laboriosos y fácilmente sujetos a errores. En cambio, pensemos en lo fácil que resulta aplicar con un ordenador, por ejemplo, algunas leyes físicas. El movimiento de un péndulo o la caída de un objeto bajo la atracción gravitatoria son experimentos clásicos, pero que implican el uso de elementos e intrumentos de medida no siempre disponibles.

Si disponemos de un ordenador con capacidades gráficas, es muy sencillo verificar dichas leyes: basta con expresar correctamente las fórmulas y utilizar una visualización gráfica lo más parecida posible al caso real. Observar como se comporta el modelo creado en el ordenador, al variar las condiciones que influyen sobre él, es casi un juego (como inciso, muchos de los recientes juegos de ordenador están basados en conceptos similares, eso sí, debidamente camuflados).

La posibilidad de reproducir en el ordenador experimentos que de otro modo resultarían imposibles o peligrosos, tiene como efecto secundario, un desarrollo de la capacidad de síntesis mental. Poder dedicarse totalmente al examen del fenómeno, sin perderse en complejos cálculos, hace más fácil captar la esencia del experimento y, a largo plazo, habitúa la mente a distinguir los factores irrelevantes de aquellos que realmente influyen en un evento determinado.

Además, la introducción de elementos aleatorios en un modelo de ordenador, resulta sumamente fácil. Sólo habrá que tener en cuenta la forma de generar diferentes tipos de variables aleatorias que se adapten a los distintos sucesos.

El programa cuyo listado presentamos a continuación, utiliza un método eficiente para generar variables con distribución normal. Te ayudará a crear modelos para un pequeño negocio.

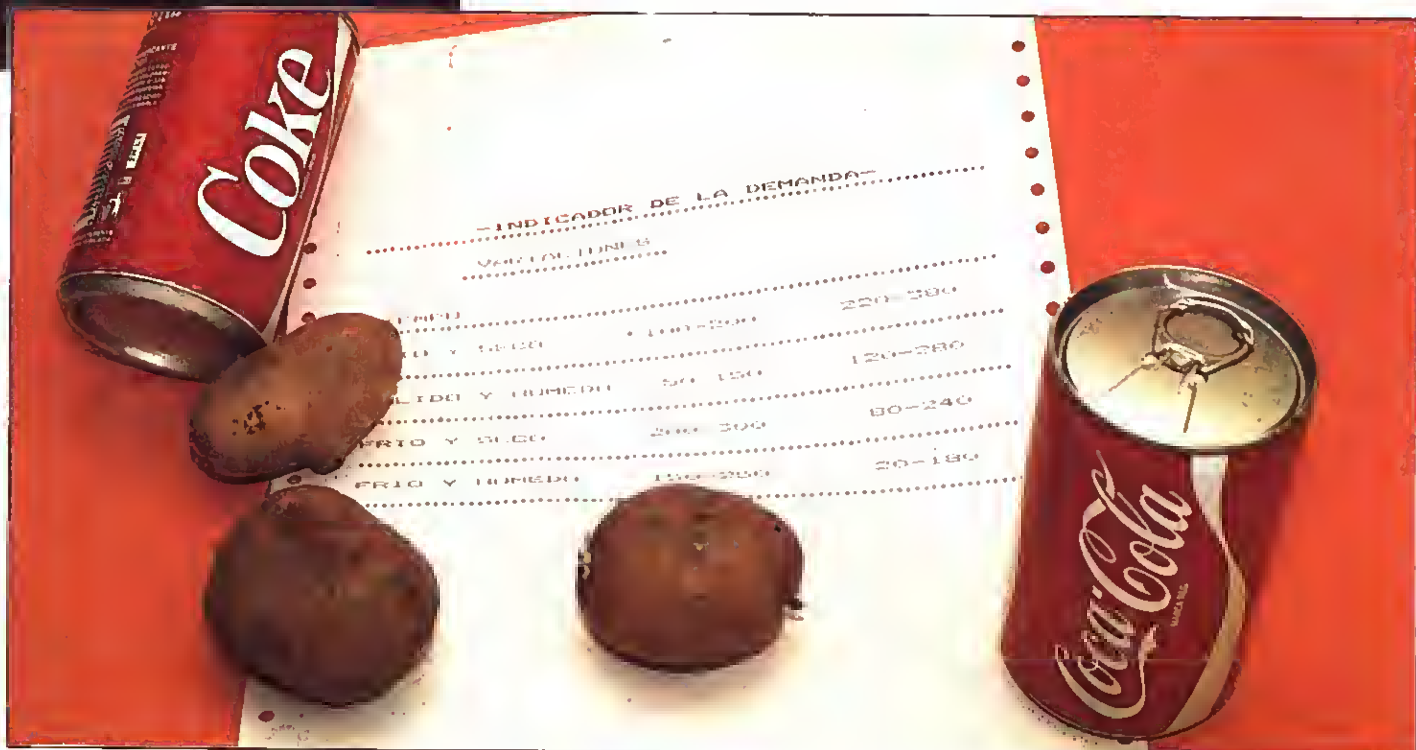
Puedes considerar este programa como un juego (bastante interesante, por cierto), pero no pierdas de vista que, en realidad, se trata de un modelo para una situación real.



```
10 CLS:COLOR15,4,4:PRINT
   "COMIDA"
20 DIM D$(10):KEY OFF
30 A$(1)="Calido y seco"
   :D(3)=150:D(4)=300
40 A$(2)="Calido y humedo"
   :D(5)=100:D(6)=200
50 A$(3)="Frio y seco"
   :D(7)=250:D(8)=160
60 A$(4)="Frio y humedo"
   :D(9)=200:D(10)=100
580 C1(1)=10:C1(2)=20
590 C2(1)=50:C2(2)=25
600 C3(1)=1:C3(2)=12
620 PRINT"Cuantos jugadores
    (1-6)?"
625 N$=INKEY$:N=VAL(N$):IF
    N<1 OR N>6 THEN 625
630 FORI=1 TO 2:FORJ=1 TO N
640 TP(I,J)=0
650 NEXT J,I
700 FOR K=1 TO 10
710 P1=.1*INT(10*(.3+RND(1)
    /2))
720 P2=.1*INT(10*(.2+RND(1)
    /2))
730 CLS:PRINT TAB(15);"Dia "
    ;K
740 LOCATE 0,4:PRINT"Probabi
    lidad de que el dia sea..
    .":PRINT:PRINT"Caluroso";
    TAB(15);100*P1;"%"
750 PRINT"Seco";TAB(15)
    ;100*P2;"%"
760 GOSUB 1400
770 U1=RND(1):U2=RND(1)
780 V1=SQR(2*LOG(1/U1))
790 V2=COS(2*3.1415*U2):
    V3=SIN(2*3.1415*U2)
800 Z1=INT(V1*V2):Z2=INT
    (V1*V3)
810 A1=P1*P2:A2=P1
820 A3=P1+P2-A1:A4=1:
    F=RND(1)
821 IF F<= A1 THEN R=1
822 IF F>A1 AND F<=A2 THEN
    R=2
823 IF F>A2 AND F<=A3 THEN
    R=3
824 IF F>A3 AND F<=A4 THEN
    R=4
830 PRINT"El tiempo es ";
    A$(R):PRINT
840 D(1)=INT(D(1+R*2)+Z1*25)
    :D(2)=INT(D(2+R*2)
    +Z2*40)
850 PRINT"Demanda de patatas
       ";D(1)
860 PRINT"Demanda de latas de
     Cola    ";D(2)
1000 PRINT:PRINT"JUG.  INGRES
     OS.  COSTES  BENEFICIO"
1010 GOSUB 1600
1020 NEXT K
```

```
1030 FOR I=1 TO 2000:NEXT I
1100 CLS:PRINT"RESULTADOS FI
     NALES AL CABO DE":PRINT
     "10 DIAS"
1110 PRINT:PRINT"JUGADOR
     BENEFICIO TOTAL"
1120 FOR J=1 TO N
1130 PRINT J,TT(J):NEXT J
1150 END
1400 PRINT:PRINT"ORDEN"
1410 FOR J=1 TO N
1420 PRINT:PRINT"Jugador ";J
1430 PRINT"Patatas asadas
     pedidas    ";:INPUT
     O(1,J)
1440 PRINT"Latas de Coca-Cola
     pedidas";:INPUTO(2,J)
1450 NEXT J:CLS
1460 RETURN
1600 FOR J=1 TO N
1610 FOR I=1 TO 2
1620 L=O(I,J)
1630 IF D(I)<L THEN L=D(I)
1650 RV(I)=C2(I)*L
1670 TC(I)=C1(I)*O(I,J)
1680 IF D(I)<=L THEN TC(I)=
     TC(I)-C3(I)*(O(I,J)-
     D(I))
1700 P(I)=RV(I)-TC(I)
1710 TP(I,J)=TP(I,J)+P(I)
1720 NEXT I
1730 TT(J)=TP(1,J)+TP(2,J)-
     2000
1740 E=RV(1)+RV(2)
1750 C=TC(1)+TC(2)+200
1760 P=P(1)+P(2)-200
1770 PRINT J;TAB(8);E
     TAB(17);C TAB(25);P
1780 NEXT J
1782 LOCATE 0,22:PRINT"Pulsa
     una tecla"
1785 T$=INKEY$:IF T$="" THEN
     1785
1790 RETURN
```

El programa se concentra en el aspecto de «nadar o hundirse» propio de un negocio: la pérdida o la ganancia. Al teclear RUN para ejecutarlo, puedes elegir entre ser el único participante o actuar junto a otros empresarios, hasta un máximo de cinco. También puedes elegir el jugar la parte de dos o incluso tres de los empresarios, tomando decisiones distintas en cada uno de los tres papeles. Puedes aprovechar la oportunidad para comparar los resultados de, por ejemplo, llevar los negocios con cautela en uno de los casos, y actuar con carácter más emprendedor y asumiendo riesgos en el otro. Sea la que fuere tu elección, al empezar la ejecución del





programa debes introducir el número de jugadores.

Cada jugador regenta un puesto que vende patatas cocidas y refrescos de cola. La demanda depende del tiempo que haga. Cuando hace frío, las patatas se venden bien. Cuando luce el sol, se venden muy bien las bebidas refrescantes. Por desgracia el empresario tiene que comprar un stock la víspera de cada día que abre su quiosco, y no conoce el tiempo que va a hacer. Afortunadamente dispone del pronóstico del hombre del tiempo, que suele ser correcto el 70 por ciento de los días. El ganador del juego es el jugador que después de 10 días de comprar y vender haya hecho las mayores ganancias.

La primera parte del programa (hasta la línea 650) establece las variables para la representación en pantalla y las condiciones del negocio. Puedes pagar una renta de 200 pesetas por día. Las patatas las compras a 10 pesetas cada una y las vendes a 50 pesetas. La Coca-Cola te cuesta 20 pesetas y la vendes a 25. Cada día de negocio te deshaces de la mercancía que te sobra a un precio de desecho: 1 peseta para las patatas y 12 pesetas para la Coca-Cola. Cada juego dura diez días.

El indicador de demanda del género es el mostrado en la tabla adjunta, y aquí es donde se introduce un importante elemento en el modelo. Como es natural, los mejores días para las patatas calientes son de los peores para las bebidas frías; sin embargo la demanda está lo suficientemente repartida entre los dos artículos como para que los empresarios hagan un razonable negocio, haga el tiempo que haga. No obstante, todo depende en alguna medida del pronóstico del tiempo, no perdiendo de vista en ningún momento que no es totalmente fiable. Las probabilidades de que haga un día caluroso y seco están determinadas por las líneas 710 y 720; posteriormente se utilizan (líneas 810 a 824) para simular el tiempo.

Las líneas 770 a 800 utilizan un método sofisticado para generar variables aleatorias con distribución normal. Después se usan dichas variables para simular la demanda en la línea 840. La línea 770 genera dos variables aleatorias (U1 y U2), pero recuerda que no son verdaderamente aleatorias. Por ello se procesan en tres fórmulas matemáticas. En la línea 780, se halla el inverso de U1 y se eleva al cuadrado, obteniendo después su logaritmo natural. Finalmente se asigna la raíz cuadrada del resultado como valor de la variable V1. La línea 790 asigna a V2 el coseno de una circunferencia de radio U2, y a V3 el seno del mismo arco. Las variables V1, V2 y V3 sufren una nueva manipulación (línea 800) para dar finalmente las variables normalmente distribuidas Z1 y Z2.

El resto del programa, desde la línea 1600 hasta el final, se dedica a la organización de la entrada de datos.

Cuando juegues este juego tienes que tener en cuenta hasta la última peseta.



¡PARTICIPA EN INPUT!



BASES

PROGRAMAS: Una vez desarrollado tu programa, que debe ser original y no haber sido enviado a ninguna otra publicación, puedes enviárnoslo aquí grabado en cassette, diskette o microdrive. Es preferible que vaya acompañado por un listado de impresora, pero no es imprescindible.

El programa habrá de venir acompañado por un texto que aclare cuál es su objetivo, el modo de funcionamiento y una explicación del cometido que cumplen las distintas rutinas que lo componen. El texto se presentará en papel de tamaño folio y mecanografiado a dos espacios. No importa que la redacción no sea muy clara y cuidada; nuestro equipo de expertos se encargará de proporcionarle la forma más atractiva posible.

ARTICULOS E IDEAS: Se aplica lo anteriormente dicho para los textos que acompañan a los programas; es decir, conviene detallar al máximo lo que desees que aparezca publicado en la revista, de la manera que te gustaría que otra persona hubiera explicado eso mismo.

UN JURADO propio decidirá en cada momento qué colaboraciones reúnen los requisitos adecuados para su publicación, y evaluará la cuantía del premio en metálico al que se hagan acreedoras.

No olvidéis indicar claramente para qué ordenador está preparado el material, así como vuestro nombre y dirección y, cuando sea posible, un teléfono de contacto. Entre todos los trabajos recibidos durante cada mes SORTEAREMOS:

- **Un premio de 50.000 ptas.**
- **Un premio de 25.000 ptas.**
- **Un premio de 10.000 ptas.**
  en material microinformático a elegir por los afortunados.

¡No os desaniméis!, por muy simples o complejas que puedan parecer vuestras ideas, todas serán revisadas con el máximo interés.

INPUT MSX
Alberto Alcocer, 46, 4.° B
28016 Madrid

NOTA: INPUT no se responsabiliza de la devolución del material que no vaya acompañado por un sobre adecuado con el franqueo correspondiente.

LOGO, LENGUAJE INTUITIVO

Cada vez son más los niños que se introducen en la informática a través del LOGO, un lenguaje diseñado pensando en el aprendizaje. Seymour Papert, **a quien se atribuye un claro conocimiento de la mentalidad infantil, ha introducido una nueva revolución en las aulas con su desarrollo de un lenguaje de alto nivel.**

En 1960 los ordenadores eran muy caros. La potencia de cálculo de tu micro habría costado millones de pesetas, ya que incluso los ordenadores más grandes no podían almacenar más de unos 144 Kbytes. Por razones de economía, los lenguajes de ordenador estaban diseñados para utilizar la menor cantidad de memoria posible, y su concepción pretendía hacerlos fáciles para el ordenador aunque así resultaran más difíciles para el programador.

Con la aparición del microordenador en los años setenta, los lenguajes de programación ganaron en popularidad, ya que los nuevos micros, al igual que los grandes ordenadores de los años sesenta, tenían memorias pequeñas. Cobró cuerpo la idea de que «sencillo para el ordenador» no significaba necesariamente «sencillo para el programador», y se fueron aceptando las dificultades de aprendizaje de lenguajes como el BASIC, como una característica de la programación.

LA ELECCION DE UN LENGUAJE

Al conectar la mayoría de los micros domésticos, éstos trabajan en BASIC (Beginners' All-Purpose Symbolic Instruction Code=Código de Instrucciones Simbólicas de Uso General para Principiantes), siendo éste el lenguaje que domina la mayoría de los propietarios de micros. Sin embargo no hay razón para que esto sea así necesariamente. El BASIC es sólo un programa en código máquina que está presente automáticamente en el ordenador, y de hecho aún existen máquinas para uso doméstico en las que el BASIC ha de cargarse desde una cinta o disco antes de empezar a programar.

Esto significa que es perfectamente posible cambiar el lenguaje que tu ordenador es capaz de entender; todo lo que tienes que hacer es cargar un programa un código máquina que le permita reconocer las instrucciones y realizar las acciones adecuadas. Tal vez ya has descubierto que es relativamente fácil ampliar el BASIC de tu máquina con comandos adicionales.

Pero también es posible no adaptarse meramente al programa existente utilizado por el intérprete BASIC, sino sustituirlo completamente.

Desde los primeros tiempos de la aparición de los ordenadores, se han desarrollado alrededor de unos cien lenguajes diferentes, para diversos fines, además de los múltiples lenguajes "de fabricación casera" diseñados para ordenadores particulares. Algunos de estos lenguajes son tan especializados en sus aplicaciones que nunca te encontrarás con ellos, excepto en los más altos niveles de investigación. Pero hay otros que por lo menos son tan prácticos para el usuario doméstico como el BASIC, y en algunos casos incluso más.

Que puedas disponer o no de un lenguaje particular depende de que puedas obtener o no el programa que permite a tu micro operar con él. Los diferentes lenguajes se suministran normalmente como cualquier otro programa, sobre cinta o disco, y en algunos casos sobre un *chip* de ROM. Su disponibilidad depende del ordenador que tengas. Por ejemplo, muchas máquinas de aplicación comercial que disponen del sistema operativo CP/M pueden elegir entre más de una docena de lenguajes, y en algunos casos entre varias versiones de cada uno. Y existe también una elección relativamente variada para la mayoría de los ordenadores domésticos.

NIVELES DE COMUNICACION

Un lenguaje es una forma de comunicar entre el ordenador y tú. Es



algo que entendéis los dos, un compromiso entre un lenguaje natural (por ejemplo el inglés y el código máquina en binario, que es con lo que realmente trabaja la máquina). Se dice que un lenguaje es de bajo nivel cuando es cercano al lenguaje propio del ordenador. Un ejemplo de ello es el lenguaje ensamblador. Los lenguajes de alto nivel, como el LOGO, pueden estar muy cerca del lenguaje natural. La próxima generación de ordenadores (llamada quinta generación) probablemente utilizará uno de dichos lenguajes para aceptar instrucciones directamente del inglés.

El BASIC se sitúa en algún punto intermedio entre los dos extremos, y según la opinión de muchos programadores no se puede decir que sea un buen compromiso, ya que ni es fácil de entender, ni es rápido para ser usado por el ordenador.

LOS COMIENZOS DEL LOGO

En 1967, un grupo de investigación del **Instituto Tecnológico de Massachusetts** (MIT), estableció unos planteamientos nuevos de acercamiento a los ordenadores. Se propusieron la creación de un lenguaje que resultara fácil para el programador aunque no fuera tan fácil para el ordenador. El resultado fue el LOGO.

El equipo estaba encabezado por **Seymour Papert,** un sudafricano expatriado. **Papert** había trabajado estrechamente con **Jean Piaget,** el famoso psicólogo infantil, según el cual los niños sólo pueden entender un concepto abstracto si se les presenta de una forma concreta. Pensaba que el aprendizaje de un niño debería tener lugar a través de sus propios descubrimientos, en lugar de adoptar una actitud pasiva ante las cosas. Estas ideas tuvieron una gran influencia en el desarrollo del LOGO.



También ejerció influencia el trabajo de **Mary Minsky,** investigadora sobre inteligencia artificial en el mismo instituto durante los años 60. La inteligencia artificial es la ciencia que se ocupa de simular con máquinas los aspectos de la inteligencia humana. Los ordenadores no son inteligentes, sino que sólo obedecen instrucciones que se les han de dar de una forma muy detallada. Los factores que hay que considerar cuando se resuelve un problema son muchos y muy variados, de modo que la simulación de un proceso de este tipo en un programa de ordenador es una tarea gigantesca. El mundo de la inteligencia artificial necesita lenguajes de programación que permitan simular el aprendizaje humano y su capacidad de tomar decisiones. El LISP es un potente lenguaje de programación desarrollado con tal objeto, y del que se hablará más adelante en INPUT. El LOGO es esencialmente un dialecto del LISP, y aunque puede manejar palabras y números, está orientado principalmente a la programación de gráficos.

El nombre de LISP deriva de «List Processing». Su estructura de datos básica no es una matriz de números ni una cadena de caracteres, sino una lista, y dado que una lista puede estar formada por símbolos o por otras listas, es fácil procesar datos no numéricos. Sin embargo el LISP no es fácil de aprender.

DEMOSTRACIONES CONCRETAS

Seymour Papert y sus colegas iban buscando una "puerta de acceso" por la que los niños pudieran entrar al mundo de la programación. Se identificaron así tres áreas de interés para los niños: gráficos, música y robótica. Los niños están interesados en realizar dibujos sobre el monitor

de su ordenador, en servirse de él para crear sonidos electrónicos, y en controlar máquinas desde el teclado. De las tres áreas, la robótica era la más excitante y sugestiva, lo que llevó a **Papert** a crear la tortuga robótica.

El desarrollo de la tortuga se vio estimulado porque en aquella época no había monitores baratos. La tortuga, un robot que se arrastra por el suelo llevando un lápiz, está controlada por el ordenador. El lápiz puede subirse o bajarse a medida que la tortuga se va moviendo, con lo que ésta puede realizar dibujos. Permite que los niños relacionen la geometría con sus propios movimientos al andar o pintar. Con frecuencia se veía a los niños que utilizaban la tortuga moviéndose y probando subrutinas antes de ordenar que lo hiciera la tortuga.

La tortuga se llamó así en honor de **Grey Walter,** un neurólogo y cibernético británico que construyó "tortugas cibernéticas" en los años 50. Se trataba de vehículos accionados eléctricamente que medían el nivel de carga de sus baterías, y cuando estaban bajas se dirigían hacia un dispositivo de carga al que se enchufaban ellas solas. La "tortuga" de **Grey Walter** fue uno de los primeros robots verdaderos.

La tortuga robótica original del MIT casi se extinguió con la aparición de los ordenadores personales y la facilidad de representar cosas en una pantalla de una forma sencilla y barata. Fue sucedida por una versión bidimensional: la tortuga de la pantalla. Se trata de un cursor, representado a veces como un *chevron* y otras veces como una pequeña tortuga. Obedece a las mismas instrucciones que la tortuga robótica, no se rompe y es mucho más barata.

Sin embargo la tortuga robótica ha hecho su reaparición en las escuelas, ya que permite que participen más niños en las actividades de programación, proporcionando un modelo para la «geometría de cuerpos», y una excitante introducción concreta a un mundo abstracto.

Estos robots se están haciendo muy populares en las escuelas elementales y están cubriendo su objetivo original de introducir a los niños en la programación de ordenadores de una forma divertida y comprensible.

HABLANDO EL LENGUAJE

¿Qué tienen que ver los juguetes mecánicos con un lenguaje de ordenador? **Papert** considera el ordenador como un vehículo para la creatividad y la expresión de ideas. Piensa que la mejor manera de aprender cosas acerca de los ordenadores es crecer en una cultura del ordenador, de la misma forma que la mejor manera de aprender italiano es pasarse una temporada en Italia. En la conferencia anual correspondiente a 1983 de la principal asociación profesional de ordenadores de América, pidió que se buscase una fórmula para darle un ordenador a cada niño americano. Piensa que es el niño quien debe programar al ordenador y no el ordenador quien programe al niño. Los medios que propone para lograr esto pasan por el LOGO, que según su punto de vista da a los niños control sobre uno de sus más potentes recursos, proporciona una base de partida para resolver problemas fuera del cálculo, y permite que se presenten de una forma sencilla e inteligible las ideas matemáticas complejas.

Papert explicó su filosofía en su célebre libro "Frenesí mental, Niños, Ordenadores e Ideas Potentes". Desde la publicación de su libro en 1980, han aparecido versiones del LOGO para la mayoría de los ordenadores domésticos. Está disponible para diversos microordenadores y la mayor parte de las versiones se parecen estrechamente al LOGO original del MIT. También hay en el mercado varios programas con nombres tipo LOGO, como el «Logo Dart» y el «Logo Graphics».

Se trata de simulaciones de la tortuga gráfica, que sólo constituye una pequeña parte del LOGO. El lenguaje tiene utilidades completas para el proceso de listas y palabras, funciones matemáticas y de sonido y muchas otras prestaciones que no aparecen en los programas "Pseudo-Logo".

El LOGO es el primer lenguaje "próximo al usuario". Como es sencillo de aprender y abunda en los colegios, se piensa con frecuencia que es "para chicos". Nada más lejos de la verdad. El profesor **Harold Abelson,** uno de los diseñadores del LOGO en el MIT, declara: "Al trabajar con el LOGO hemos descubierto algunas cosas importantes."

"Un lenguaje de ordenador puede ser sencillo y potente al mismo tiempo. De hecho no se trata de aspectos conflictivos sino complementarios, ya que es la falta de potencia expresiva en los lenguajes primitivos tales como el BASIC, lo que hace tan difícil para los principiantes el escribir programas simples que hagan algo interesante. Lo que es más importante, hemos visto que es posible dar a la gente control sobre potentes recursos informáticos, que pueden usar como herramientas aprendiendo, jugando y explorando."

El LOGO es también un lenguaje que crece. La universidad de Edimburgo está desarrollando una versión llamada Control-LOGO, que permita un control más sofisticado de los robots. **Papert** pretende que las futuras versiones incluyan "mundos" en los que los niños puedan jugar con las ideas de la Física, de la misma forma que juegan con la Geometría con la Tortuga Gráfica.

PROGRAMANDO EN LOGO

Al cargar el LOGO desde un *cassette* o un *chip* de ROM, te enviará un mensaje de este tipo:

```
BIENVENIDO AL LOGO

?
```

La interrogación (?) es un anunciador o invitación a que introduzcas algo. El LOGO está esperando a que se le dé alguna orden. Si tecleas:

```
HOLA LOGO, TE HE ESTADO
 BUSCANDO
```

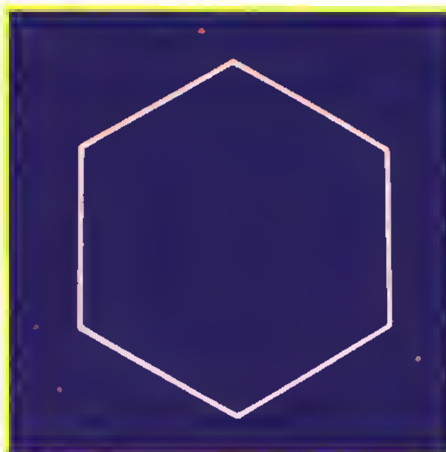
El LOGO responderá con:

```
NO SE COMO HACER HOLA
?
```

Ha mirado la primera palabra sin identificarla como instrucción, y te ha informado de ello.

Un comando que sí reconocerá es ST, que quiere decir mostrar la tortuga (Show Turtle). Llamará a la tortuga desde las profundidades del ordenador hasta la pantalla. La forma de la tortuga aparecerá en una dirección particular. Este es su encabezamiento, aparecerá provista de un lápiz y listo para pintar.

La instrucción para mover la tortuga hacia adelante es FORWARD, que se puede abreviar a FD. FORWARD es una instrucción del LOGO que requiere una entrada. Hay que decirle a la tortuga cuánto tiene que moverse. Si se pulsa FORWARD 100, la tortuga se moverá 100 unidades hacia adelante. Dejará tras de sí una línea. Si está utilizando una tortuga que se desplaza por el suelo, por ejemplo el Valiant, se moverá 100 centímetros en la dirección que tenga de frente.









Si pulsas:

```
FORWAR1ØØ
```

El LOGO responderá

```
NO SE COMO HACER FORWAR1ØØ
```

Esto se debe a que, al no haber intercalado un espacio entre el comando y la cantidad numérica, el LOGO considera FORWARD100 como una palabra diferente de FORWARD, y dado que FORWARD100 no forma parte de su vocabulario envía un mensaje de no reconocimiento. Lo mismo ocurriría si en vez de FORWARD hubieras introducido FERWARD.

Hay un sencillo editor de línea que permite corregir los errores antes de pulsar RETURN. Las teclas del cursor permiten trasladarse a lo largo de la línea hasta llegar a la falta. La tecla de borrado (delete) elimina el carácter situado a la izquierda del cursor. Para insertar nuevos caracteres no tienes más que teclearlos. El texto que hay más a la derecha se moverá automáticamente para dejar sitio al texto insertado. Algunas versiones de LOGO permiten recuperar y editar una línea después de haber pulsado RETURN.

La instrucción BACK trabaja de la misma forma que FORWARD. Su abreviatura es BK. Hace que la tor-

tuga se mueva hacia atrás, y, como antes, requiere una entrada numérica. Puedes dar cualquier número como entrada para FORWARD o BACK. Así cambiará la posición de la tortuga, pero no su orientación.

Para girar la tortuga, utiliza LEFT y RIGHT. Se pueden abreviar por LT y RT. Al igual que FORWARD y BACK, requieren una entrada numérica. Por ejemplo, RIGHT 39 hará girar la tortuga 39 grados a la derecha. Análogamente LEFT 123, hará que la tortuga gire 123 grados hacia la izquierda.

ALGUNOS JUEGOS SENCILLOS

Si se utiliza el LOGO con niños, es mejor no decirles que RIGHT 96 hace girar la tortuga 96 grados hacia su derecha. Hay que dejar que experimenten con diferentes valores numéricos de entrada y descubran por sí mismos el efecto de los diferentes valores. Se pueden ensayar diversos juegos encaminados a que descubran los valores de diferentes ángulos.

Se puede marcar una posición en la pantalla y ver cuántas instrucciones necesita un niño para detener la tortuga debajo de la marca. O se puede dibujar una carretera en la pantalla y hacer que el niño haga avanzar la tortuga, perdiendo un punto por cada instrucción y un punto cada vez que la tortuga se salga de la carretera. Más adelante daremos en este artículo un procedimiento para dibujar una carretera.

El programa se puede almacenar en disco o cinta, y cargarse cada vez que los niños quieran jugar con él. Si quieres jugar inmediatamente, dibuja la ruta directamente sobre la pantalla del televisor, utilizando un lápiz de cera, o un rotulador de los usados para pintar sobre pizarra blanca. Se puede borrar con facilidad al terminar. De esta forma es más fácil dibujar rutas más complicadas.

Hay una variedad de juegos que se pueden practicar con una tortuga de suelo, basados en recorrer laberintos, tirar cosas, recoger objetos en diversos puntos, etc. Se puede jugar a la tortuga empujona, una variante del juego de los bares, que utiliza una tortuga en vez de una moneda.

Es posible jugar a la tortuga empujona en la pantalla, escribiendo un procedimiento para dibujar la tortuga, y colocándola al principio.

MAS COMANDOS

Para jugar con las instrucciones de traslación y giro se requieren algunas otras instrucciones de LOGO o **primitivas.** Si la pantalla no está en modo "enrollado", es decir, si la tortuga desaparece por la parte superior pero no reaparece por la inferior, necesitas hacer que vuelva a aparecer.

HOME hará que la tortuga regrese a su posición y orientación originales en el centro de la pantalla.

También necesitas poder borrar la pantalla para hacer nuevos dibujos.

La instrucción CLEARSCREEN, que se abrevia CS, borrará todos los dibujos y situará la tortuga en su posición HOME.

A veces desearás que la tortuga se mueva sin dibujar. La instrucción PENUP, abreviada PU, levanta el lápiz de la tortuga. Si utilizas una tortuga de suelo, se levanta el lápiz que lleva en la panza. La tortuga de la pantalla simplemente deja de dibujar.

La instrucción PENDOWN, abreviada PD, hace que baje el lápiz de la tortuga de suelo y que la tortuga de pantalla dibuje de nuevo.

Las instrucciones CLEARSCREEN, HOME, SHOWTURTLE, PENUP y PENDOWN son primitivas de LOGO que no requieren entradas.

He aquí un ejemplo ilustrativo sobre estas primitivas. Al final de cada línea has de pulsar la tecla RETURN.

```
SHOWTURTLE
LEFT 45
FORWARD 71
RIGHT 135
PENUP
FORWARD 50
PENDOWN
LEFT 45
BACK 71
PENUP
HOME
PENDOWN
```

Si quieres intentar alguna otra cosa, puedes borrar el dibujo de la pantalla utilizando la instrucción CLEARSCREEN.

CONSTRUCCION DE UN PROCEDIMIENTO

Hasta aquí todas las actividades descritas han sido del modo 'inmediato". Has estado hablando directamente a la tortuga, que ha ejecutado inmediatamente tus órdenes, igual que un pelotón de soldados obedeciendo a un sargento instructor en una plaza de armas. Hay otro modo de funcionamiento en LOGO, el modo de procedimiento.

En el modo de procedimiento se nombra una serie de comandos, los cuales se escriben después del nombre. De esta forma el nombre asignado al procedimiento pasa a formar parte del vocabulario del LOGO. La tortuga responderá a este nombre ejecutando los comandos que figuran en su definición. Para definir un procedimiento, utiliza TO seguido del nombre elegido. Se puede utilizar cualquier nombre excepto el de una primitiva de LOGO existente. Las cosas resultan más fáciles si asignas a los procedimientos nombres que definan su función.

Se entenderá mejor esto si ponemos un ejemplo. He aquí un procedimiento que enseña a la tortuga a dibujar un zig-zag:

```
TO ZIGZAG
FORWARD 20
LEFT 150
FORWARD 20
RIGHT 150
FORWARD 20
LEFT 150
FORWARD 20
RIGHT 150
FORWARD 20
LEFT 150
FORWARD 20
RIGHT 150
END
```



PRESENTA...



SENSACIONALES PROGRAMAS EN CARTUCHO Y CASSETTE

FLIGHT PATH 737.



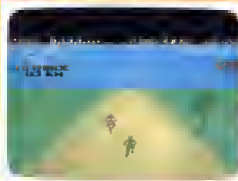
Colócate a los mandos de un jet comercial. Disponemos de control total sobre los mandos del avión, y puedes escoger entre 6 niveles de dificultad.
P.V.P.: CART. 3.490 pts.
CASS. 1.900 pts. 32K.

FRUITY FRANK



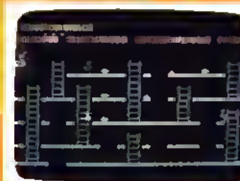
Tu jardín ha sido invadido por monstruos de fruta madura. La única forma de combatirlos es lanzarles fruta fresca del jardín.
P.V.P.: CASS. 1.900 pts. 32K.

SPARTAN X



Son muchos los peligros que te acechan. Ten los reflejos bien despiertos, pon tus fuerzas en estado de alerta, y a luchar.
P.V.P.: CASS. 1.900 pts. 32K.

CHUCKIE EGG



Debes recoger los huevos antes de que nazcan los pollitos y se coman el maíz. Pero ojo con el Pato Loco.
P.V.P.: CASS. 1.900 pts. 32K.

NIGHT FLIGHT



Con tu pequeño avión, debes ir dando luz a la noche, hasta que el cielo esté de nuevo azul. Date prisa en realizar tu misión, de lo contrario..
P.V.P.: CART. 2.900 pts.
CASS. 1.900 pts. 16K

STAR AVENGER



Imagina el juego de batalla más rápido que jamás hayas visto. Piensa además, en los más excitantes gráficos y sus 5 niveles de dificultad. Todo ello es Star Avenger.
P.V.P.: CASS. 1.900 pts. 32K.

GYRO ADVENTURE



Ponte a los mandos de tu helicóptero y combate a los enemigos que se enfrentan a ti. Podrás mover el helicóptero en todas direcciones, mantenerlo en el aire y disparar. **P.V.P.: CART. 2.900 pts.**
CASS. 1.900 pts. 16K.

SUPER CROSS FORCE



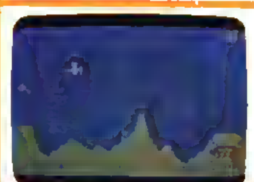
Sólo queda una esperanza para la supervivencia ante el ataque de los malvados Marpul. Tú podrás atacarles, con tus naves dispuestas en paralelo o en diagonal.
P.V.P.: CART. 2.900 pts. 16K.

JUMP LAND



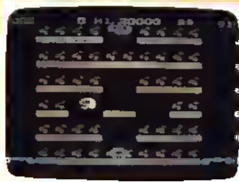
Tu mayor obsesión han sido siempre los pasteles, y por ello, te has visto envuelto en situaciones complicadas que has salvado gracias a tus reflejos.
P.V.P. CASS. 1.900 pts. 16K.

ROGER RUBBISH



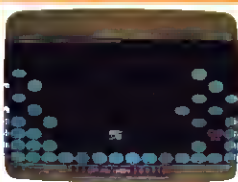
Los perversos contaminadores de planetas están llenando nuestra galaxia de residuos nucleares. Roger Rubbish es el más famoso recogedor de basuras espaciales.
P.V.P.: CART. 2.900 pts. 16K.

FRUIT PANIC



Un día, Walky, para divertirse se fue al país de las galas. ¿Cuánta fruta podrá comerse Walky?
P.V.P.: CASS. 2.000 pts. 16K

DIZZY BALLOON



En este mundo hay seres voladores y atacan cuerpo a cuerpo. Si los haces explotar, se irá abriendo el cielo y tendrás la oportunidad de escapar.
P.V.P.: CASS. 2.000 pts. 32K.

CASTLE COMBAT



El castillo galáctico, ha caído bajo la dominación de los Tyranes. Tu nave STAR DUSTER, está preparada para el combate ¿Te atreves?
P.V.P.: CART. 2.900 pts. 16K.

NICK NEAKER



Cuando estás dormido, muchas cosas suceden a tu alrededor. Algunos objetos de tu casa toman vida, como en el caso de la zapatilla NICK.
P.V.P.: CART. 2.900 pts.
CASS. 1.900 pts. 16K.

CHAMP



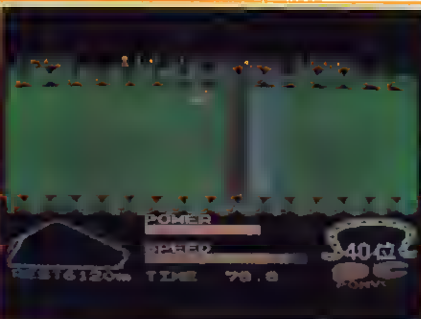
Champ es un completo Ensamblador/Monitor para tu MSX. Champ te permite escribir y trazar programas en código máquina con el mínimo esfuerzo.
P.V.P.: CART. 3.890 pts.
CASS. 2.400 pts. 32K.

KARATE



Has conseguido entrar en la cueva de los piratas y ahora comienzan tus problemas. Los murciélagos gigantes, moradores de estas cuevas pueden chuparte la sangre. Cuando te encuentres con los piratas, deberás enfrentarte a ellos con tu depurado estilo de Kárate **P.V.P.: CART. 3.490 pts. CASS. 1.900 pts. 32K.**

GRAND NATIONAL



Si te gustan las carreras de caballos, no te quedes como un espectador, participa. Ahora puedes correr con tu caballo, en la más prestigiosa carrera del mundo, el GRAND NATIONAL.
P.V.P.: CASS. 2.000 pts. 32K.

ENVIENOS A **MICROBYTE**
P.º Castellana, 179, 1.º - 28046 Madrid

Nombre
Apellidos
Dirección
Población
D.P. Teléfono

ENVIOS GRATIS

JUEGO	Cart	Cass	Precio	TOTAL

PRECIO TOTAL PESETAS

Incluyo talón nominativo ☐
Contra-Reembolso ☐

Pedidos por teléfono **91 - 442 54 33 / 44**

Después de teclear TO ZIGZAG seguido de RETURN la invitación o *prompt* cambia de ? a >. Esto te indica que estás en modo procedimiento. Al terminar el procedimiento, pulsa END y el anunciador volverá a ser ?, lo que significa que nuevamente estás en modo inmediato.

ZIGZAG forma ahora parte del vocabulario del LOGO. Si lo tecleas, la tortuga dibujará un zig-zag.

El LOGO tiene una instrucción de repetición que se puede utilizar para teclear varias veces la misma cosa. Se podría escribir de nuevo ZIGZAG como:

```
TO ZIGZAG
REPEAT 3 [FORWARD 20 LEFT 150
  FORWARD 20 RIGHT 150]
END
```

La rutina que ha de repetirse aparece encerrada en paréntesis cuadrados, precedida por REPEAT y el número de veces que ha de ser repetida. Si piensas en un círculo como en una serie de traslaciones y giros cortos, REPEAT facilita el dibujo de curvas. Por ejemplo REPEAT 180 [FORWARD 1 RIGHT 1] dibujará un semicírculo.

PONIENDOLO TODO JUNTO

Ya tienes todos los ingredientes para dibujar una pista de carreras. La manera de resolver un problema con el LOGO es dividirlo en lo que **Seymour Papert** llama "Bytes a medida de la mente". Así, para empezar con el borde interior, dibuja primero una curva:

```
REPEAT 180 [FORWARD 1
RIGHT 1]
```

Y ahora una línea recta:

```
FORWARD 100
```

Puedes combinar dos curvas y dos bordes rectos en un procedimiento para dibujar el interior de la pista:

```
TO INTERIOR
REPEAT 2 [FORWARD 100 REPEAT
  180 [FORWARD 1 RIGHT 1]]
END
```

Es perfectamente legítimo tener en REPEAT dentro de otro REPEAT siempre que te acuerdes de cerrar todos los paréntesis al final.

Para el exterior de la pista, necesitas una curva más amplia, creada aumentando el tamaño de los pasos de la tortuga.

```
REPEAT 180 [FORWARD 2
  RIGHT 1]
```

es demasiado grande, pero algo cercano a:

```
REPEAT 90 [FORWARD 3 RIGHT 2]
```

estará bastante bien.

Vayamos ahora al exterior de la pista.

```
TO EXTERIOR
REPEAT 2 [FORWARD 100 REPEAT
  90 [FORWARD 3 RIGHT 2]]
END
```

Pero si tecleas:

```
EXTERIOR
INTERIOR
```

no tendrás el resultado ideal.

Puedes escribir un procedimiento para empezar a dibujar en un sitio más adecuado, mediante un procedimiento llamado BEGIN:

```
TO COMENZAR
PENUP
LEFT 40
FORWARD 110
RIGHT 130
PENDOWN
END
```

Otro procedimiento moverá la tortuga a una posición adecuada para dibujar la parte interior de la pista, proporcionando además una línea de partida.

```
TO MOVER
RIGHT 90
FORWARD 30
LEFT 90
END
```

Para ver la pista debes llamar a todos los procedimientos a la vez, tecleando:

```
COMENZAR
EXTERIOR
MOVER
INTERIOR
```

Ahora puedes escribir un procedimiento que coloque a la tortuga en la línea de salida:

```
TO PRINCIPIO
LEFT 90
FORWARD 15
RIGHT 90
END
```

Una vez que un procedimiento está ya en la memoria de un ordenador, el LOGO te permite usarlo de la misma forma que cualquier primitiva de LOGO. Esto significa que puedes usar los procedimientos para ayudar a definir otros nuevos procedimientos. Así, puedes combinar los bloques constructivos que has llamado COMENZAR, EXTERIOR, MOVER, INTERIOR y PRINCIPIO, en un nuevo procedimiento que denominaremos JUGAR:

```
TO JUGAR
COMENZAR
EXTERIOR
MOVER
INTERIOR
PRINCIPIO
END
```

Cada vez que tecleas JUGAR, aparecerá la pista con la tortuga en la línea de salida.

Todos los programas de LOGO se construyen de esta forma. La fragmentación de un programa en pequeños bloques constructivos hace más fácil su construcción y depuración.

En un próximo artículo veremos cómo realizar programas más complicados utilizando las técnicas aquí descritas.

INPUT JUEGOS

COLECCIONABLE DE PROGRAMACION



Aquí aprenderás cómo diseñar tus propios juegos para ordenador; desde los más simples a los más complejos de estrategia e inteligencia. Una gran abundancia de ejemplos son la manera más agradable de introducirte en los secretos de la programación.

MOVIMIENTO Y ANIMACION

- LOS PRINCIPIOS DE LA ANIMACION
- MOVIMIENTO DE GRAFICOS
- COMO UTILIZAR LOS GRAFICOS INCORPORADOS

¿Quieres darle vida a tus juegos de programación? Empieza entonces con estos sencillos caracteres gráficos, que puedes generar a partir de la memoria gráfica de tu ordenador.

Jugar con los juegos que se venden para tu ordenador es divertido sólo hasta cierto punto. Llega un momento en que la mayor parte de la gente siente la necesidad de dar rienda suelta a su imaginación y crear programas de juegos propios.

La programación de juegos no es fácil; tienes que empezar con cosas muy sencillas e ir aumentando poco a poco en complejidad. Pero eso te ayudará a pensar con lógica y aumentará tu habilidad como programador. Y también te divertirás más.

Lo primero que tienes que aprender en la programación de juegos, aparte de los trucos del BASIC, es la técnica de la animación.

Para crear la ilusión de movimiento, el programador de ordenadores utiliza en gran parte la misma técnica que el caricaturista que anima una película de dibujos. Lo que hace es crear dos (o más) imágenes y alternarlas rápidamente (idealmente, unas 24 veces por segundo).

Pero existe una diferencia importante. En la animación de dibujos, el dibujante cuenta con el proyector de películas, que le permite olvidarse de una imagen cuando ya no la necesita. En la animación por medio de un ordenador no ocurre esto. Como no hagas algo para evitarlo, cualquier segmento de imagen que «proyectes» sobre un área dada de la pantalla permanecerá allí indefinidamente.

Una forma de olvidarse de la imagen que ya no se necesita es, simplemente, imprimir algo sobre ella. El ordenador no puede situar dos imágenes al mismo tiempo en la misma posición de la pantalla.

Así si, por ejemplo, la línea 10 de un programa cualquiera dice al ordenador que imprima una A en una determinada posición, cualquier otra línea de programa que imprima (por ejemplo) una B en la misma posición se desembarazará de la A.

Los programas que siguen contienen varios ejemplos de esta clase de sustitución.

¿Pero qué ocurre si no tienes nada



que imprimir encima del carácter no deseado? Acuérdate en ese caso de incluir en alguna línea posterior una instrucción que imprima un espacio en blanco en la correspondiente posición de la pantalla.

Si te olvidas de este detalle, tu pantalla pronto se verá abarrotada con trozos no deseados de brazos, piernas y cuerpos.

La manera de obtener los caracteres gráficos en la pantalla difiere mucho de un ordenador a otro. Hay diferencias entre los caracteres gráficos incorporados en ROM y lo mismo ocurre con la forma en que se imprimen (con PRINT) en pantalla. Por último también hay diferencias en el modo de hacer que se muevan.

Esta versión de un ciempiés para **MSX** utiliza exactamente los mismos signos de máquina de escribir que otras máquinas, pero el método de generarlos sobre la pantalla es diferente. Intenta introducir el siguiente programa:

```
)))     )))     )))     )))
ooo<    ooo<    ooo<    ooo<
)))     )))     )))     )))
```

```
10 CLS
20 PRINT")))"
30 PRINT"ooo<"
40 PRINT")))"
50 LOCATE0,0
60 PRINT"((("
70 PRINT"ooo<"
80 PRINT"((("
90 GOTO10
```

Cuando lo ejecutes (RUN) observarás una imagen rápidamente cambiante. Se debe a los conjuntos separados de símbolos de las sentencias PRINT que se superponen unos a otros. Al mismo tiempo, la sentencia GOTO de la última línea crea un bucle continuo: le dice al ordenador que vuelva a empezar.

Sin las líneas 10 y 50 el programa no podría funcionar adecuadamente. En ambas líneas se hace uso de instrucciones de movimiento de cursor y de borrado de pantalla, que combinadas con la instrucción PRINT permiten llevar a cabo la animación.

¿Cómo funcionan? La instrucción LOCATE 0,0 de la línea 50 coloca el cursor en la esquina superior izquierda de la pantalla (línea 0, columna 0). Esto quiere decir que toda actividad de impresión posterior comienza en esta posición, de modo que todos los caracteres que se imprimen, desde la línea 60 en adelante, lo hacen encima de los que ya estaban allí.

Por su parte la instrucción CLS de la línea 10 hace algo más. Después de hacer volver al cursor a la parte superior izquierda de la pantalla, borra todo su contenido, dejándola preparada para que aparezca la siguiente imagen.

MAS DESPACIO

El movimiento del insecto es hasta ahora más bien rápido y tal vez haya que retardarlo. La manera más fácil de hacer esto es utilizar un bucle FOR...NEXT. Introduce pues la línea:

```
45 FOR T=1 TO 50:NEXT
```

Cuando pulses RETURN y ejecutes (RUN) el programa, el movimiento resultará mucho más pausado. El bucle FOR...NEXT actúa como un contador, en este caso contando hasta 100, antes de que el programa vaya a la línea 50.

Puedes cambiar la duración de la



pausa sin más que sustituir «100» por cualquier otro número de tu elección: cuanto más grande el número, mayor será el retardo.

Intenta también cambiar la posición del bucle de retardo FOR...NEXT a la línea 15. Dependiendo del retardo que escojas, habrá una apreciable pausa —una pantalla limpia— cuando se ha borrado una imagen pero todavía no ha sido sustituida por la siguiente. Por esta razón es mejor utilizar LOCATE 0,0 que CLS, dentro de un programa de este tipo.

Aunque la imagen ya está ahí, todavía resulta algo espasmódica, debido a que el bucle de retardo sólo actúa sobre la primera imagen. Se puede obtener un movimiento mucho más adecuado introduciendo en el programa otro bucle FOR...NEXT que actúe sobre la segunda imagen. Inserta pues:

```
85 FOR I=1 TO 50:NEXT
```

Este retardo es más corto, a fin de crear un movimiento «de pierna» ligeramente irregular, pero naturalmente, puedes cambiarlo si quieres.

CARACTERES EN MOVIMIENTO

El siguiente paso es alterar el programa de forma que el «cuerpo» del insecto parezca cruzar la pantalla. Para ello el BASIC MSX nos ofrece dos instrucciones que vamos a utilizar. La primera de ellas, que puede ser adecuada en aplicaciones sencillas, es la instrucción TAB. TAB siempre va seguido de un número entre paréntesis, por ejemplo TAB (15), que hace que el cursor se coloque en la columna 15 de la pantalla, o por una variable también entre paréntesis.

Además TAB siempre forma parte de la sentencia PRINT a la que se aplica. En este caso se hace uso de una variable, la variable P de la línea 10, para hacer que la posición de TAB se mueva por la pantalla. Esta variable cambia su valor al ir incluida dentro de un bucle FOR...NEXT.

```
10 FOR P=0 TO 32
20 CLS
25 PRINT TAB(P)")))"
30 PRINT TAB(P)"ooo<"
40 PRINT TAB(P)")))"
50 FOR I=1 TO 45:NEXT
55 LOCATE0,0
57 PRINT TAB(P)"((("
60 PRINT TAB(P)"ooo<"
70 PRINT TAB(P)"((("
80 FOR I=1 TO 50:NEXT
90 NEXT  P
```

Lo que has hecho es suprimir la sentencia GOTO original de la línea 90. Ahora ha sido remplazada por un bucle FOR...NEXT, que incrementa en 1 la variable P cada vez que se repite el programa. Como P forma parte de la sentencia TAB, el insecto se mueve sobre la pantalla, a razón de un paso por cada ciclo del programa.

Al ejecutar (RUN) el programa, verás que el insecto se mueve sobre la pantalla y se para al llegar al lado de-



recho. Para que la acción comience de nuevo necesitas agregar:

```
100 GOTO 10
```

La otra instrucción es LOCATE X,Y que permite colocar el cursor en la columna X, fila Y de la pantalla. LOCATE se utiliza normalmente antes de la sentencia PRINT. En ese caso se ha utilizado LOCATE con el valor P para la columna, y los valores 7,8 y 9 para la fila. Estos tres valores corresponden a cada una de las partes de nuestro insecto, que se imprimen en filas consecutivas. La utilización de LOCATE en este ejemplo tan sencillo resulta casi más complicada que la de TAB. Pero hay que pensar que TAB sólo permite desplazar el cursor dentro de una línea y no de una línea a otra. Por ello, en caso de movimientos más complejos siempre será preferible usar LOCATE.

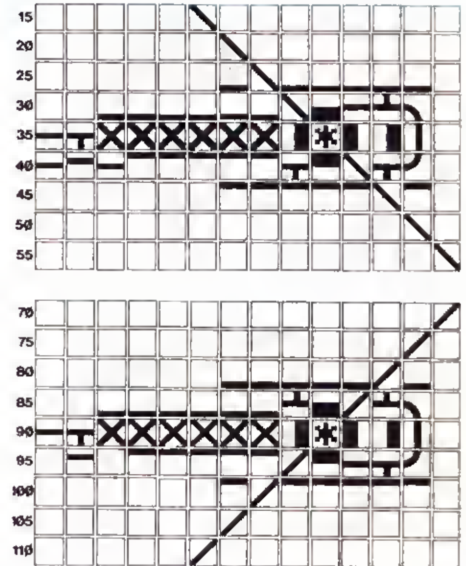
```
10 FOR P=0 TO 32
20 CLS
25 LOCATEP,7:PRINT")))"
30 LOCATEP,8:PRINT "ooo<"
40 LOCATEP,9:PRINT ")))"
50 FOR I=1 TO 45:NEXT
57 LOCATEP,7:PRINT "((("
60 LOCATEP,8:PRINT "ooo<"
70 LOCATEP,9:PRINT "((("
80 FOR I=1 TO 50:NEXT
90 NEXT  P
```

GRAFICOS EN ROM

El insecto nos ha ayudado a comprender algunos de los fundamentos de la animación, pero como gráfico resulta bastante simple. Para conseguir algo más elaborado podemos utilizar los caracteres gráficos de la memoria ROM. Todos los **MSX** disponen de estos caracteres gráficos a los que se puede acceder desde el teclado. Para ello no hay más que pulsar la tecla GRPH, o bien las teclas SHIFT y GRPH simultáneamente. En la pantalla habrán aparecido numerosos símbolos gráficos que podrás combinar a tu gusto. Para que te hagas una idea de lo que puedes conseguir, observa el gráfico del helicóptero. Está hecho utilizando exclusivamente caracteres gráficos. Aquí tienes el programa que lo dibuja. Si quieres ver al helicóptero en acción teclea y escribe RUN.

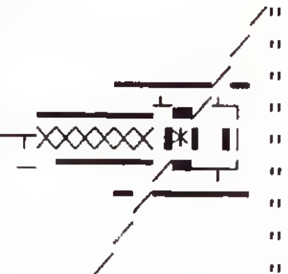
```
10 CLS
12 PRINT"
14 PRINT"
16 PRINT"
18 PRINT"
20 PRINT"
22 PRINT"
24 PRINT"
26 PRINT"
28 PRINT"
```





Cómo construir un helicóptero. Cada cuadro es un gráfico de ROM.

```
30 FOR D=1 TO 50:NEXT
32 LOCATE 0,0
34 PRINT"
36 PRINT"
38 PRINT"
40 PRINT"
42 PRINT"
44 PRINT"
46 PRINT"
48 PRINT"
50 PRINT"
52 FOR J=1 TO 50:NEXT
54 GOTO 10
```





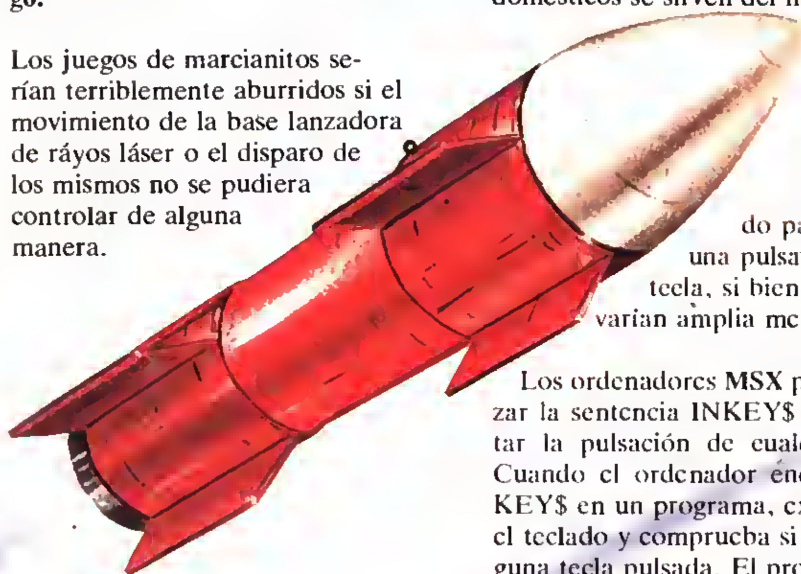


DERECHA...IZQUIERDA... ARRIBA...¡FUEGO!

- DETECCION DE LAS PULSACIONES DE TECLA
- LANZAMIENTO DE MISILES
- CONTROL DE UN GRAFICO MOVIL

Los juegos de extraterrestres se basan en la habilidad del jugador para controlar los sucesos de la pantalla. He aquí cómo controlar el movimiento, lanzar misiles e integrarlos en un juego.

Los juegos de marcianitos serían terriblemente aburridos si el movimiento de la base lanzadora de rayos láser o el disparo de los mismos no se pudiera controlar de alguna manera.











El control por teclado de ese tipo de movimientos es una importante faceta de todos los juegos de marcianitos, por lo que si piensas escribir alguno, es importante comprender sus principios.

Para ello, el primer paso es hacer que el ordenador reaccione cuando pulses una tecla.

DETECCION DE LAS PULSACIONES DE TECLA

En principio todos los ordenadores domésticos se sirven del mismo método para detectar una pulsación de una tecla, si bien los detalles varían ampliamente de unos a otros.

Los ordenadores MSX pueden utilizar la sentencia INKEY$ para detectar la pulsación de cualquier tecla. Cuando el ordenador encuentra INKEY$ en un programa, explora el teclado y comprueba si hay alguna tecla pulsada. El programa que sigue es un ejemplo típico de cómo se utiliza INKEY$ para hacer esto.

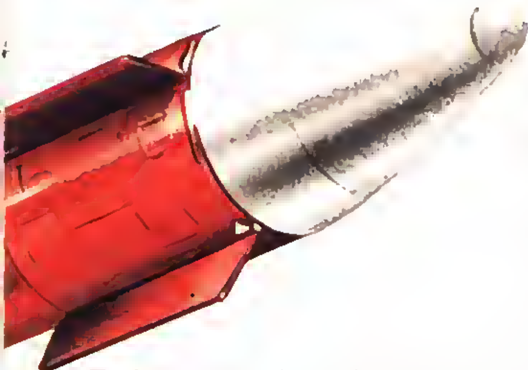
```
20 CLS
30 A$=INKEY$:IF A$="" THEN 30
40 LOCATE14,10:PRINT"AUUG!"
```

Haz correr el programa una y otra vez (con RUN) y comprobarás que pulsando cualquier tecla, excepto SHIFT, GRPH, CTRL y CODE, se origina la aparición en pantalla del mensaje AUGG!. El programa trabaja así:

La línea 20 limpia la pantalla. La línea 30 hace que el ordenador espere hasta que se pulse una tecla antes de continuar con el programa. Date cuenta que no está incluido un espacio entre las comillas. Por ello, la línea 30 significa: «Si INKEY$=nada, o si no ha sido pulsada tecla alguna, vuelve a comprobar». Es importante disponer del IF...THEN 30, por-

que de otra forma el ordenador comprobaría solamente una vez si ha sido presionada una tecla y sólo durante una fracción de segundo.

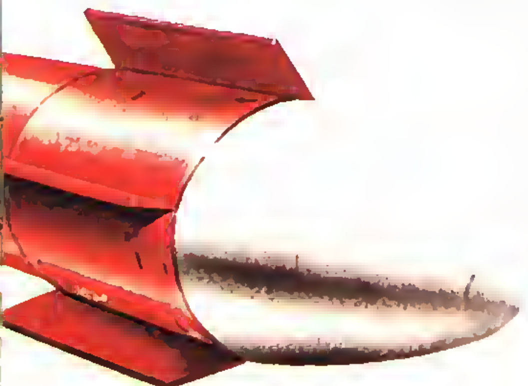
Tan pronto como la tecla sea pulsada, A$ toma el valor de esa tecla. Por ejemplo, si presionamos el 3, entonces A$=«3». Esto es suficiente para que la línea 40 visualice «AUUG!» en la pantalla.



En la mayoría de los juegos debes presionar una determinada tecla para mover un tanque, una nave espacial o cualquier otra cosa. Si alteras la línea 40 verás cómo se consigue esto.



```
20 CLS
30 A$=INKEY$:IF A$=""THEN 30
```



```
40 IF A$="d" THEN PRINT
   "ESTUPENDO":STOP
50 LOCATE14,10:PRINT"AUUG!"
```

La línea 40 comprueba si la tecla «d» ha sido pulsada; en otras palabras ¿es A$ igual a d? Si no es así, tu MSX ignorará la línea 40 y continuará en la 50 (trata de descubrir por qué es necesario el STOP). En este programa hay una cosa más de importancia. La «d» debe escribirse entre comillas. De otra forma el ordenador podría confundirla con una variable.

LANZAMIENTO DE UN MISIL

Vamos a utilizar lo que hemos aprendido en un programa que lanza un misil cuando se pulsa la tecla «f». Este es el listado:

```
20 CLS:KEY OFF
30 LOCATE 15,22:PRINT"*^*"
40 A$=INKEY$:IFA$=""THEN 40
50 IF A$<>"f" THEN 40
55 Y=21
60 LOCATE 16,Y:PRINT"^"
70 Y=Y-1
75 FOR J=1 TO 10:NEXT
80 LOCATE 16,Y+1:PRINT" "
90 IF Y>0 THEN 60
```

En la línea 20 se limpia la pantalla y se eliminan los códigos de las teclas de función (KEY OFF). La línea 30 dibuja la base de lanzamiento de misiles. Al llegar a la línea 40 el programa espera hasta que se pulsa una tecla. En la línea 50 se comprueba si la tecla pulsada es la «f». Si no es así, el programa vuelve a la línea 40 y queda esperando una nueva pulsación. En la línea 60 se dibuja el misil, que acaba de ser lanzado, en la fila Y de la pantalla. La primera vez Y vale 21 pero a medida que el misil asciende, Y se decrementa en una unidad. Esto tiene lugar en la línea 70. La línea 80 se encarga de borrar la posición que ocupaba anteriormente el misil. Por último, en la línea 90 se comprueba si el misil ha alcanzado la parte superior de la pantalla (en cuyo caso la coordenada Y vale 0) para terminar el programa.

MOVIENDOSE POR LA PANTALLA

El programa de la base de misiles, tal como está, es más bien aburrido, pero si se dota a la base de movimiento, las cosas mejoran un poco. Veamos cómo puede moverse la base.

```
20 P=15
30 CLS:KEY OFF
40 LOCATEP,22:PRINT"*^*"
50 A$=INKEY$:IF A$=""
   THEN 50
60 IF A$="i" THEN P=P-1:
   GOTO90
70 IF A$="d" THEN P=P+1:
   GOTO90
80 GOTO 50
90 IF P>33 THEN P=33
100 IF P<1 THEN P=1
110 GOTO30
```

La variable P de la línea 20 establece la posición inicial de la plataforma lanzamisiles, que queda dibujada en la pantalla al ejecutarse la línea 40. Entre las líneas 50, 60 y 70 se encuentra distribuida la rutina de lectura del teclado que comprueba si se ha pulsado la tecla I o la tecla D.

Al pulsar I, se resta uno de la variable P con lo que la base de misiles se desplaza a la izquierda. Pulsando D, la base se desplaza hacia la derecha al incrementarse en uno el valor de P.

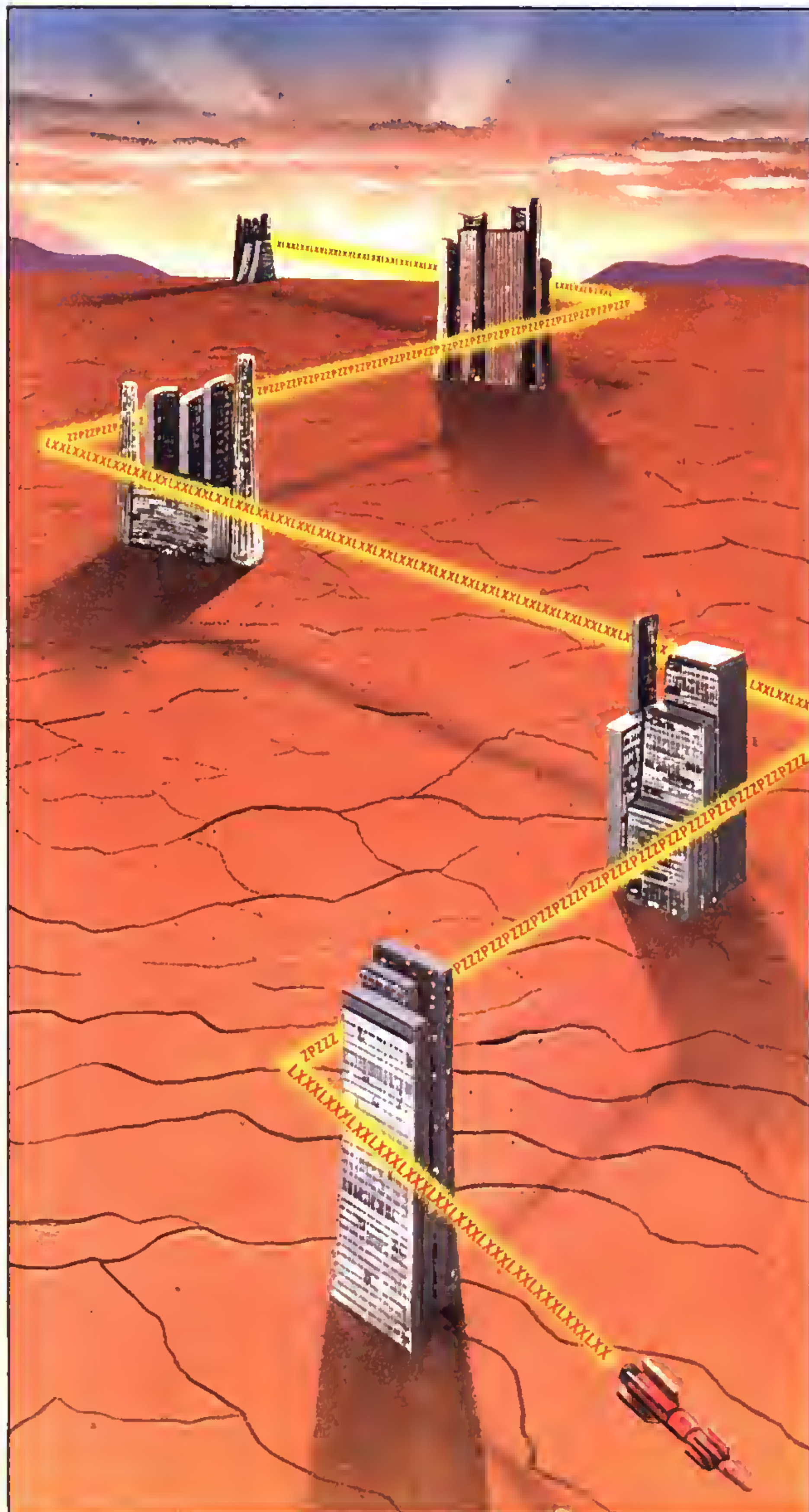
El GOTO de la línea 80 hace que el programa vuelva a la línea 50 si se pulsó cualquier otra tecla. Las líneas 90 y 100 limitan los valores de P para evitar que la base de misiles pueda salirse de la pantalla.

Finalmente la línea 110 hace que el programa vuelva a la línea 30 y pueda dibujarse la plataforma lanzamisiles en su nueva posición.

CREA TU PROPIO JUEGO

Ahora que ya dispones de algunos conocimientos y de algún bloque constructivo puedes acometer la realización de un juego sencillo que utilice estos bloques. Teclea el programa que sigue y escribe RUN.

```
10 KEY OFF:COLOR15,12,12
20 P=15
40 CLS
```



```
50 A=RND(-TIME)
60 A=INT(RND(0)*30)+3
70 LOCATEA,1:PRINT"*"
80 LOCATEP,22:PRINT" *^* "
90 A$=INKEY$:IF A$=""
   THEN 90
95 IF A$="z" THEN P=P-1
100 IF A$="x" THEN P=P+1
105 IF P>33 THEN P=33
110 IF P<1 THEN P=1
115 IF A$="f" THEN P1=P+2:
    D=21:GOTO130
120 GOTO 80
130 LOCATEP1,D:PRINT"^":
    D=D-1
140 FOR I=1 TO 10:NEXT
150 LOCATEP1,D+1
160 PRINT" "
170 IF D>1 THEN 130
180 IF P1=A THEN 40
200 GOTO 80
```

Verás aparecer una estrella cerca de la parte superior de la pantalla. Con las teclas Z e Y podrás trasladar la plataforma lanzamisiles hacia la derecha y hacia la izquierda hasta situarla bajo la estrella. Pulsa entonces la tecla F para lanzar un misil y destruir la estrella.

El programa está compuesto de tres secciones: hasta la línea 80, de la 90 a la 120, y las líneas 130 a 200 son análogas a las del anterior programa de lanzamiento de misiles. Se han cambiado las variables y el GOTO, pero lo único nuevo es la línea 180. En ella se comprueba si la estrella y el misil están en la misma vertical. Si ocurre esto el programa vuelve a empezar.

La sección central, líneas 90 a 120, es una versión resumida del movimiento por la pantalla que vimos anteriomente.

La primera sección del programa hasta la línea 80, realiza varias funciones. Las líneas 50, 60 y 70 generan un valor aleatorio que representa la posición de la estrella. Por su parte la línea 80 establece la posición de partida de la plataforma de misiles dibujándola a continuación.

Cuando hayas tecleado el programa escribe RUN y procura afinar tu puntería.

MODELOS DE IRREGULARIDAD

- FIGURAS GEOMETRICAS
- MEDICION DE UN OBJETO
- AUTOSIMILITUD
- FRACTALES Y GRAFICOS
- DIBUJAR DRAGONES

Utiliza tu microordenador para explorar el fascinante mundo de la geometría fractal: la herramienta matemática que contribuye a explicar la irregularidad que da forma al mundo real.

¿Cuanto mide una cuerda? La respuesta es fácil de encontrar si la cuerda está extendida, porque entonces basta con medirla. Lo mismo ocurre si la cuerda está enredada, pero ahora resultará un poco más difícil de medir, ya que se trata de un objeto irregular.

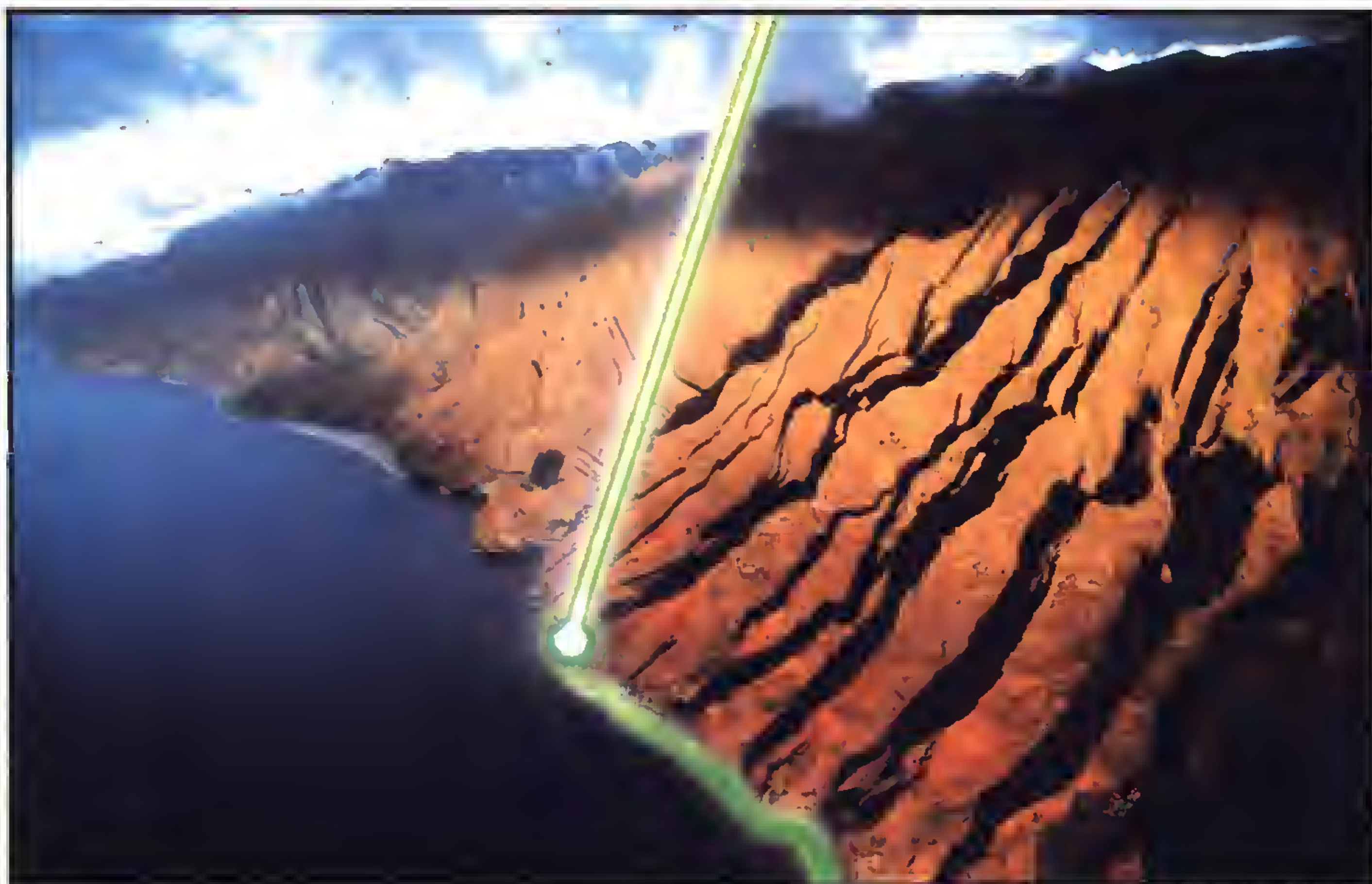
Naturalmente, cualquiera que tenga un poco de sentido común estirará la cuerda antes de medirla. Pero ¿y si tiene que medir algo de forma compleja y que no se pueda desenredar?

Imagina un trozo de costa de cinco kilometros de longitud formada por acantilados. ¿Cómo lo ves en tu imaginación? Los cinco kilómetros ¡son iguales para el vuelo del cuervo, o son los que tendrías que andar si fueras siguiendo el contorno real de la costa? La diferencia es considerable y parece evidente que una ruta que recorra todas las irregularidades será notablemente más larga que el camino en línea recta.

Pero ¿cuánto más largo? Supongamos que llevas una cinta métrica muy larga y muy flexible y que decides medir todos los detalles. Ante todo, tendrás que incluir las principales irregularidades de la costa, como las entradas de los ríos. Pero, hay irregularidades menos importantes también, como las salientes de las rocas. Al avanzar con la cinta métrica deberás incluir aún pequeños detalles como las rocas aisladas, o los granos de arena. Y si pudieras verlo más detalladamente, observarías que los propios granos de arena tienen una superficie irregular cuyo perímetro se puede medir.

Puede parecer un ejercicio absurdo: después de todo ¿quién necesita conocer las distancias con tanto detalle? Pero, inidica un hecho importante: cuando se trata de elementos cuyo contorno o superficie no es perfectamente uniforme, la medición depende realmente del tamaño del instrumento de medida.

La ciencia tradicional utiliza mode-



los para las curvas y superficies que se suponen uniformes. Y cuanto más se amplian estas formas, más planas parecen, de la misma forma que —en una aproximación grosera— la forma de la Tierra es la de una esfera, pero tan grande que para los que la habitan la superficie parece plana.





No obstante, como demuestra el ejemplo de la costa, algunos objetos no parecen planos ni siquiera cuando están muy ampliados. Y en la naturaleza hay numerosos ejemplos de objetos que poseen una estructura detallada en muchas escalas distintas. Pero sólo recientemente los investigadores y matemáticos han reconocido que son dignos de estudio y han obtenido un modelo para su estructura en una nueva serie de figuras geométricas llamadas **fractales**.

La palabra fractal proviene de una palabra latina que quiere decir «irregular». Fue acuñada por el matemático **Benoit Mandelbrot**, que identificó a las fractales como una nueva generación de objetos matemáticos adecuados a la representación de un modelo de objetos naturales irregulares.

La diferencia entre una fractal y una curva suave es espectacular en apariencia, pero es igualmente espectacular desde el punto de vista teórico. La mayor parte de las personas están familiarizadas con la idea de las tres dimensiones del espacio; una superficie tiene sólo dos dimensiones y una línea sólo una. Y para explicar algunas teorías físicas de su época, **Albert Einstein** propuso una cuarta dimensión: el tiempo. Las fractales amplían la noción de las dimensiones de forma notable al exigir dimensiones de números fraccionarios en lugar de enteros.

Esto no quiere decir que una fractal exija direcciones independientes de uno y medio o de dos y medio. Media dirección es algo que no tiene sentido ni para un matemático. En cambio, se puede concebir un nuevo tipo de dimensión derivado de la forma en que se comporta la fractal ampliada. Este nuevo tipo de dimensión de las respuestas previstas de 3, 2 ó 1, para el espacio, una superficie o una curva, pero también tiene sentido para una fractal.

MEDICION DE UN OBJETO

Si tomamos dos trozos de cuerda de la misma longitud y los colocamos uno a continuación del otro, podemos imaginar que hemos sacado una copia del trozo original, pero del doble de tamaño. Si en cambio tomáramos un cuadrado de papel, se necesitarían tres cuadrados iguales, además del primero, cuatro en total, para formar un cuadrado del doble del tamaño. Para un cubo de queso se necesitarían ochos cubos. Y si existieran los quesos de cuatro dimensiones se precisarían 16 porciones idénticas para duplicar el tamaño de uno de los trozos.

Los números 2, 4, 8 y 16 forman parte de la secuencia que se obtiene multiplicando el 2 por sí mismo varias veces: 2, 2×2, 2×2×2 y 2×2×2×2. Esta secuencia se puede representar también por: 2^1, 2^2, 2^3, 2^4, etc. Y las potencias a las que se eleva 2 —1, 2, 3, 4, etc.— son las dimensiones del objeto.

Un objeto unidimensional (la cuerda) debe multiplicarse por 2, uno bidimensional (el papel) se multiplicará por 4, y así sucesivamente. Ahora, supongamos que se encontrara un objeto que precisara de tres partes iguales para duplicar el tamaño de una de las partes. Este número se halla entre dos (que corresponde a una dimensión) y cuatro (dimensión 2), de modo que se podría decir que la dimensión del objeto está entre 1 y 2.

A primera vista podría parecer una idea improbable, aunque desde el punto de vista lógico es evidente. Pero resulta que esta teoría logra explicar el efecto de aumento que se descubre al tratar de medir una fractal.

Es incluso posible hacer un diagrama de un elemento que funcione de

esta forma. El matemático alemán **von Koch** fue uno de los primeros en hacerlo cuando inventó la curva llamada copo de nieve, que se parece a un copo de nieve infinitamente arrugado. Cada una de sus caras está formada por una copia de sí mismo, de un tercio de su tamaño y la dimensión es de apenas un poco más de 1,26. En la naturaleza, una línea costera característica puede considerarse como de dimensiones muy similares.

La curva fractal se define muy exactamente, mientras que en el mundo natural las variaciones son mucho mayores, de modo que como en el caso de muchos modelos matemáticos la situación artificial sólo ofrece una aproximación a la naturaleza. No obstante, brinda a la ciencia la mejor explicación de numerosas estructuras naturales desde las venas y arterias del cuerpo hasta las formas de las montañas, pasando por los meandros de los ríos y la corteza de los árboles.

AUTOSIMILITUD

Como podrás observar si vuelves al ejemplo de la cuerda, el papel y el queso que citábamos antes, el objeto se formó a partir de ejemplares más pequeños de sí mismo. Un bloque grande de queso está formado por varios bloques más pequeños de queso de la misma forma. Se dice entonces que el objeto es similar a sí mismo, porque si observamos una parte cúalquiera del objeto nos encontramos con que el objeto parece una versión reducida de sí mismo.

En las fractales matemáticas este principio se aplica muy rígidamente, por eso son tan regulares. Las fractales naturales no son exactamente idénticas a sí mismas, de modo que si se aumenta una sección de costa, o un trozo de corteza, no se corresponde exactamente con la estructura de la cual proviene. No obstante, a grandes rasgos parece igual, como si se tratara de una porción de costa de otro lado o de una corteza de otro árbol. Este fenómeno recibe el nombre de autosimilitud estadística y es muy común en la naturaleza.

FRACTALES Y GRAFICOS

Bueno ¿qué tiene todo esto que ver con los ordenadores? La respuesta es que, de la misma forma que en muchos otros tipos de modelos matemáticos, las fractales se pueden emplear para programar un ordenador de modo que produzca un modelo de realidad que participa de algunas de las propiedades del mundo real. Las fractales, específicamente, encuentran sus principales aplicaciones en los gráficos obtenidos por ordenador, pues ofrecen la respuesta más práctica a los problemas creados por la producción de formas irregulares realista para cosas tales como montañas, mares y una cantidad de paisajes imaginarios.

Las fractales no sólo proporcionan el mejor modelo matemático inventado hasta ahora, sino que son ideales también para programar. El principio de autosimilitud significa que se puede construir la forma creando el mismo elemento una y otra vez, proceso que se aplica a programas sencillos y repetitivos.

Para producir una fractal matemática basta con tomar una forma simple y repetirla varias veces a distintas escalas, añadiendo gradualmente más y más detalles al dibujo. El ejemplo más sencillo se obtiene con una línea recta y aplicando una regla según la cual cada línea recta se sustituirá por un par de líneas en ángulo recto.

Si intentas dibujarlo verás que cada paso añade un nuevo nivel de detalle, ya que cada línea del dibujo se descompone en dos nuevas líneas. Y gradualmente va surgiendo el modelo. El primer programa permite que el ordenador te lo demuestre. Como todos los procesos fractales, el dibujo podría seguir indefinidamente, añadiendo más y más detalle, pero de ese modo se llegaría a detalles ínfimos que no se podrían distinguir en la pantalla del televisor. Por lo tanto, el programa ha sido pensado para que se detenga una vez ha alcanzado un determinado nivel de repetición.





```
10 CLS:COLOR15,12,12
20 MN=2
30 PI=3.1415:C=PI/180
40 L=100:X=80:Y=40:AN=PI/2:
   XX=X:YY=Y
50 SCREEN 2:GOSUB1000
60 GOTO60
1000 L=L/1.414
1010 IF L>=MN THEN 1020
1012 L=L*1.414:X=X+(L*SIN
     (AN)):Y=Y-(L*COS(AN))
1015 LINE(X,Y)-(XX,YY):XX=X:
     YY=Y:RETURN
1020 AN=AN+PI/4:GOSUB 1000
1030 AN=AN-PI/2:GOSUB 1000
1040 AN=AN+PI/4:L=L*1.414:
     RETURN
```

Ejecuta el programa e irás viendo una figura con forma de C, que va apareciendo gradualmente en la pantalla. El programa comienza dibujando una línea, después la sustituye por el primer par de líneas en ángulo recto. La subrutina situada entre las lí-





neas 1000 a 1040, que utiliza las funciones seno (SIN) y coseno (COS) para calcular los ángulos requeridos, es reclamada por la línea 50 y llamada por sí misma repetidamente dentro de un proceso recursivo, que sustituye a las líneas rectas por un par en ángulo recto.

La variable de la línea 20 es ajustada a la longitud de la línea más corta, por lo que este programa repite hasta alcanzar este valor.

Prueba cambiar el valor de la línea 20 para ver el efecto. Valores más pequeños incrementan el número de niveles recursivos, por lo que se incrementa el tiempo de ejecución. Pero si introduces números mayores, el programa se acelera y puedes observar los detalles de cómo se desarrolla la curva.

Si observas con cuidado este proceso de crecimiento, podrás ver cómo el proceso de autosimilitud significa que cualquier sección de la curva parece una versión más pequeña de la propia curva. Incluso así, no es fácil predecir la forma final de la curva.

DIBUJAR DRAGONES MATEMATICOS

Como sucede con frecuencia con las fractales, y especialmente las que se dibujan por repetición, bastan unas pocas líneas de programa para producir incluso formas complicadas. Teclea y ejecuta el siguiente programa para ver surgir una curva de dragón, que es otro ejemplo de la técnica de curva en C.

En el programa anterior se sustituía cada línea por dos líneas en ángulo recto en el mismo sentido, siempre del mismo lado de la línea. En este programa, no obstante, las dos líneas en ángulo recto están situadas primero de un lado de la primera línea, luego del lado contrario de la línea siguiente, y así sucesivamente, hasta terminar la forma.

```
5 CLS:COLOR 12,10,10
10 SCREEN 2:S=45000!
20 MN=1
25 PI=3.1415
30 C=ANT(1)/45
40 FOR I=0 TO 7:S(I)=SIN(A)
50 C(I)=COS(A):A=A+PI/4:
   NEXT I
60 L=100:X=60:Y=90:T=-1:POKE
   S,T+1:S=S-1
65 XX=X:YY=Y
70 GOSUB 1000
80 GOTO80
1000 L=L/1.414
1010 IF L>=MN THEN 1020
1012 L=L*1.414:X=X+(L*C(I)):
     Y=Y-(L*S(I)):LINE(XX,YY)
     -(X,Y):XX=X:YY=Y
1014 RETURN
1020 I=(I+T)AND7:POKES,T+1:
     S=S-1:T=1:GOSUB1000:
     S=S+1:T=PEEK(S)-1
1030 I=(I-2*T)AND7:POKES,T+1:
     S=S-1:T=-1:GOSUB 1000
1040 S=S+1:T=PEEK(S)-1:
     I=(I+T)AND7:L=L*1.414:
     RETURN
```

Además de los distintos elementos fractales que emplea, las principales diferencias entre este programa y el anterior están relacionadas con la velocidad de ejecución y la manera en que se dibuja el elemento. En lugar de calcular los ángulos cada vez que se traza un elemento, el nuevo programa calcula el seno y el coseno de los ángulos de una vez —líneas 40 y 50— luego los almacena en dos series. Extraer los valores de una serie es mucho más rápido que tener que calcularlos cada vez que se sustituye una línea, por consiguiente el programa se ejecuta más rápidamente.

Se podría aducir que la curva C y de dragón son poco más que curiosidades matemáticas. Como tales, son atractivas a la vista y fascinantes para estudiar y una vez que empiezas a modificar los valores fijados en los listados anteriores, podrás producir una gama de variaciones interesantes partiendo de las formas básicas. El segundo artículo sobre fractales te mostrará cómo introducir la incertidumbre en el dibujo de formas matemáticas para crear modelos más parecidos a los naturales.

MODELOS EXPERIMENTALES

En el citado artículo podrás ver que los principios estudiados aquí se pueden utilizar para ampliar el repertorio de gráficos.

La diferencia más importante entre las fractales producidas matemáticamente estudiadas aquí y las formas que se encuentran en la naturaleza consiste en que las formas naturales poseen elementos de desorden e irregularidad.

La siguiente serie de programas muestra cómo se pueden añadir factores aleatorios a la producción de fractales que imitan estas variaciones naturales. Y a fin de que puedas explorar los efectos del empleo de distintas formas básicas y de variables, existe también un generador de fractales de objetivo múltiple que te permitirá obtener tus propios valores esenciales. Cada vez que lo utilices obtendrás una forma distinta.

ENVIA MENSAJES SECRETOS

La codificación y el cifrado, que unen el mundo de la antigua Grecia con el mundo de la microelectrónica, son susceptibles de ser eficazmente manejados por los programas de ordenador.

La mayoría de la gente se sirve de los códigos en su vida diaria. Cuando alguien pide un tornillo del número 8 o un balón de fútbol del tamaño 5, está utilizando un código. Análogamente, las ecuaciones aparentemente incomprensibles de un científico nuclear representan la utilización de un código que permite expresar brevemente relaciones complicadas. Esto se podría hacer también utilizando palabras corrientes, pero, con frecuencia, requeriría mucho más trabajo.

En todos los anteriores ejemplos el interés reside en mejorar la comunicación de los datos, más que en buscar el secreto de la información; otra de las razones para el de uso códigos es ahorrar dinero o espacio de almacenamiento. Por ejemplo, una empresa puede redactar contratos que utilicen cláusulas y frases estándar. Si el almacenamiento se va a hacer en cinta o disco, se puede ahorrar una gran cantidad de valioso espacio codificando con números las secciones que se repiten más a menudo. A esto se le llama comprensión de los datos.

Análogamente, las respuestas estándar utilizadas en juegos de aventuras, como «The Hobbit» o «Valhalla» se codifican para ahorrar espacio de almacenamiento.

Los griegos inventaron la ciencia de enviar mensajes secretos, de forma que no es sorprendente que el nombre utilizado normalmente para la codificación —criptografía— se derive de dos palabras griegas: *Kryptos* (secreto) y *graphos* (escribir). Los términos **código** y **cifra** realmente tienen un significado ligeramente diferente, relacionado con las dos maneras en que pueden enviarse los mensajes. Cuando la información se transfiere letra por letra, se habla de cifrado. Si en cambio se transforman palabras completas o grupos de palabras en otras palabras o números utilizando un diccionario especial, a esto se le llama codificación. En la práctica, el término código se utiliza para referirse a códigos y cifras.

CODIGOS SECRETOS

La codificación y decodificación de mensajes secretos fué en otra época un área confinada principalmente a las actividades militares, o al menos a los Servicios de Inteligencia. Sin embargo hoy, el amplio uso de las líneas telefónicas públicas y de los canales de transmisión de datos comerciales importantes, ha aumentado la necesidad de la codificación.

En los albores de la revolucion informática, se utilizaban los comptómetros de IBM para romper los códigos de la Segunda Guerra Mundial. Desde entonces, todo nuevo avance en la tecnología de los ordenadores ha sido ansiosamente seguido por los maestros de espías y los reventadores de códigos. Actualmente, con una buena programación y la cifra correc-



ta, un ordenador doméstico puede funcionar como cualquier máquina convencional de criptografía. Este es el primero de dos artículos que mostrarán cómo utilizar el ordenador para producir mensajes secretos codificados, siguiendo varios métodos diferentes que, como el propio espionaje, se sitúan a niveles diferentes de sofisticación.

Aunque no seas un agente internacional, los métodos seguidos tienen interés por sí mismos, y siempre podrás servirte de ellos para enviar mensajes cifrados a otros amigos que tengan ordenador. De hecho, en este artículo hay un mensaje escondido en alguna parte.

CODIGOS DE DISTANCIA

Como su nombre sugiere, se trata de un código que está basado en la dis-



tancia de un símbolo particular a un punto dado.

Este tipo de código fué utilizado hace más de dos mil años por el general griego **Lisandro**. Las distancias de las muescas a partir de la hebilla del cinturón de uno de sus esclavos componían un mensaje secreto que ayudó al general a derrotar al Imperio Persa.

Se puede formar sencillamente un código de distancia, poniendo las letras del alfabeto en la primera línea de un papel cuadriculado y componiendo el mensaje como se muestra en la pantalla. Mientras se ve la clave de las letras en la línea superior, el mensaje es fácil de entender. Sin embargo, si se suprime esta clave, no es fácil decodificar la información.

Para hacer el descifrado todavía más intrincado, haz una rotación de la clave de las letras en la primera línea de la página. Se puede, por ejemplo, empezar por la N y cuando se llega a la Z, simplemente empezar otra vez por la A. A esto se le llama rotación cíclica.

El programa funciona creando un bucle y sirviéndote de la facilidad de LEN para evaluar el valor equivalente ASCII de cada letra del texto en claro (líneas 55, 120). La mayoría de los ordenadores incorporan una función ASCII que permite representar las letras por medio de números.

Después de convertir el mensaje en una serie de números, es fácil codificarlo utilizando una transformación lineal inmediata. Por ejemplo, en el programa, la letra V se traduce al equivalente ASCII de V menos 26 (línea 100).

Sólo falta utilizar la función TAB para imprimir el asterisco a la distancia adecuada a partir del lado derecho de la pantalla (línea 90) y el proceso de codificación está completo.

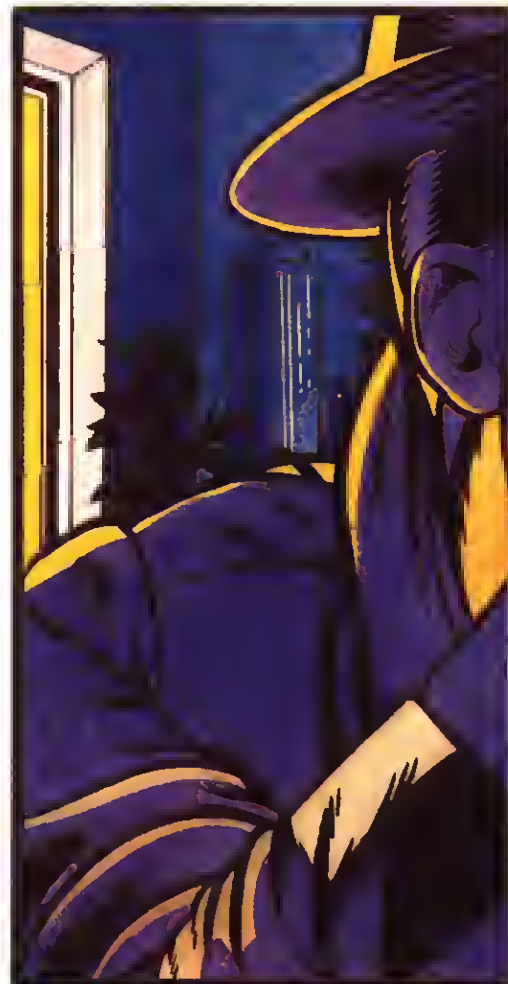
```
5 CLS:KEY OFF
10 PRINT TAB(8);"CODIGO DE
   DISTANCIA":PRINT
20 PRINT"NO DEJES ESPACIOS"
30 PRINT"ENTRE PALABRAS"
35 PRINT:PRINT"ESCRIBE EN
   MAYUSCULAS"
40 PRINT:PRINT"Cual es tu
   mensaje>":INPUT A$
50 CLS
55 FOR J=65 TO 90:PRINT CHR$
   (J);:NEXTJ:PRINT
60 FOR J=1 TO LEN(A$)
70 B$=MID$(A$,J,1)
80 V=ASC(B$)-45
90 IF V<=25 THEN PRINTTAB(V);
   "*":GOTO 120
100 V=V-26
110 PRINT TAB(V);"*"
120 NEXT J
```

UTILIZACION DEL CODIGO

Aunque el código de distancia pueda parecer demasiado sencillo para ser eficaz, tiene algunos factores a su favor. En primer lugar, antes de poder descifrar con éxito un código, hay que darse cuenta de que realmente éste existe. Y como es muy fácil enmascarar una sucesión de puntos (o asteriscos) aparentemente aleatoria en un dibujo, por lo demás inocente, hay bastantes probabilidades de que un mensaje codificado de este tipo pase inadvertido.

Durante la última guerra mundial, los agentes alemanes utilizaron este truco. Una inspección detenida a la inocente imagen de un jardín, reveló que las pinzas de la ropa en una cuerda de tender componían un mensaje secreto.

Una manera de hacer el código de distancia aún más difícil de romper es reestructurar el programa de forma que la clave de las letras sea realmente aleatoria. Un experto consciente de que tiene ante sí un código de distancia necesita ensayar, como mucho, 26 combinaciones antes de resolver el



problema. Sin embargo, si el orden de las letras de la clave es aleatorio, el número de combinaciones posibles aumenta enormemente.

EL CIFRADO DE ST. CYR

Los romanos tomaron el relevo de los griegos como maestros en criptografía. **Julio César** inventó un método de cifra por sustitución directa en el que cada letra se sustituía por la que estaba situada en el alfabeto tres lugares a la derecha. Así la A se convierte en D, la B se convierte en E, etc. Al final del alfabeto, la X se convierte en A, la Y en B y la Z en C. Utilizando este método el mensaje VIENE EL ENEMIGO se convierte en ZLHPH HN HKHOLJR.

Como se verá más adelante, el código de **César** es un caso especial del llamado cifrado de **St. Cyr**, y el resultado anterior se puede comprobar fácilmente ejecutando el siguiente programa y utilizando como número clave 3333333.

Con la caída del Imperio Romano cesaron los desarrollos en criptografía, y a pesar del creciente uso de códigos en los siglos XVI y XVII, hasta el siglo XIX no se produjeron mejoras significativas del código de **Julio César**, realizadas por la academia militar de **St. Cyr**. El cifrado de **St. Cyr** es sorprendentemente sencillo. Se compone con tres alfabetos en una escala deslizante. El alfabeto de abajo es el del texto en claro y sus equivalentes en cifra se toman de la línea superior. Partiendo de un punto de origen predeterminado, la palabra INPUT se transforma en AFHML. Una de las ventajas del cifrado de **St. Cyr** es que cada letra se puede codificar utilizando un alfabeto equivalente distinto. Esto hace muy difícil reventar un código.



En el programa de cifrado de **St. Cyr**, se ha incorporado esta utilidad por medio de un número clave, para obtener una seguridad extra. Todo el que tenga acceso a un listado de programa no podrá resolver el código, a menos que conozca también los siete números secretos.

```
10 KEY OFF:CLS
30 PRINT TAB(10);"CIFRA DE ST
   .CYR":PRINT
```



```
40 LOCATE 10,6:PRINT
   "** AVISO **"
50 PRINT:PRINT"NO DEJES
   ESPACIOS ENTRE PALABRAS"
55 PRINT:PRINT"ESCRIBE EN
   MAYUSCULAS"
60 PRINT:PRINT
70 PRINT"PULSA  1 SI DESEAS
   CODIFICAR"
80 PRINT"PULSA -1 SI DESEAS
   DECODIFICAR"
90 INPUT S
100 INPUT"ESCRIBE TU MENSAJE"
    ;A$
110 FOR A=1 TO 300:NEXTA:CLS
120 INPUT"ESCRIBE 7 NUMEROS"
    ;N$
130 FOR A=1 TO 300:NEXTA:CLS
140 FOR K=1 TO LEN(A$)
150 L=K-INT(K/7)*7+1
160 T=ASC(MID$(A$,K,1))
    +(S*VAL(MID$(N$,L,1)))
170 IF T>90 OR T<65 THEN
    T=T-(S*26)
180 PRINT CHR$(T);
190 NEXT K
```

Este programa opera de una manera muy parecida al programa de códigos de distancia. Después de leer cada letra del texto en claro, se convierte

primeramente en el equivalente numérico del carácter ASCII correspondiente. Luego se utiliza la función VAL para incrementar este número en una cantidad que depende del número deducido de su clave númerica (línea 160). Después de una comprobación para asegurarse de que el resultado está dentro de márgenes aceptables (línea 170), se aplica la utilidad CHR$ para sacar el mensaje codificado.

En los primeros días de junio de 1944 se podría haber codificado el siguiente mensaje: TROPASCAPTURADASENELPUENTEPEGASO. Utilizando el programa con el número clave 3821105, el texto codificado se convierte en: BTPQAXFIRUVRFGIUFOEQSCGOUEUHOCTP. Utilizando una variable indicadora S, que puede tomar los valores +1 ó −1, se puede utilizar el mismo programa para la decodificación. El número clave de siete cifras depende de la elección personal. Sin embargo, como estas cifras se confían frecuentemente a la memoria, es una buena práctica recurrir a algún dato familiar, tal como algún número de teléfono.

Como hay diez millones de valores diferentes para el número clave, el cifrado de **St. Cyr** es realmente de decodificación muy laboriosa. No obstante, se puede introducir una dificultad adicional codificando dos veces, es decir, poniendo el mensaje ya codificado como entrada al programa codificador por segunda vez.

Considérese el mensaje MARTINEZ ES UN ESPIA. Y utilizando los números clave 2501000 y 2506900, la secuencia cifrado-descifrado es la siguiente:

Texto	Número clave	
MARTINEZESUNESPIA		
	2501000	Codifica
RASTINGEETUNEUUIB		
	2506900	Codifica
WAYCINIJEZDNEWZIH		
	2506900	Decodifica
RASTINGEETUNEUUIB		
	2501000	Decodifica
MARTINEZESUNESPIA		

CODIGO MORSE

Incluso el más seguro de los códigos es poco útil si no se puede tansmitir rápidamente. **Napoleón** hizo algunos intentos para resolver el problema de la velocidad construyendo torres de señales por toda Francia. Para pasar mensajes de una torre a otra se utilizaba un semáforo primitivo.

Sin embargo, lo que realmente aceleró la transmisión de mensajes secretos fué la invención del telégrafo eléctrico y la cifra de sustitución conocida por código **Morse** estableció un código en que las letras estaban sustituidas por puntos y rayas.

El programa del código Morse vale para cifrar texto, o decodificar una serie de puntos y rayas a sus letras adecuadas. Por ejemplo, la conocida señal de socorro SOS se convierte en /.../---/.../ y el texto ESCAPA A MEDIANOCHE pasa a ser /./.../-.-./.-/.--./.-/.-/--/./-../../.-/-./---/-.-./..../././

Prueba los dos fragmentos de texto en el siguiente programa. Actuando en sentido inverso, si ejecutas el programa e introduces --0--0---000--0-- ***** aparecerá en la pantalla del televisor la palabra MAYDAY. Se utilizan los asteriscos para indicar al ordenador que se ha completado el mensaje.

En la primera parte de este programa se establece una cadena de caracteres con 130 puntos y rayas, para representar el equivalente Morse del alfabeto en orden secuencial (línea 250). Aunque el signo menos es un buen símbolo para representar una raya, hemos preferido utilizar un cero como símbolo gráfico para el punto, en lugar del punto ortográfico.

Se ha previsto un campo de cinco caracteres para cada letra, cuando bastaría con cuatro para todo el alfabeto. Se ha hecho esto para dar la oportunidad de ampliar el programa a quien desee introducir números que tengan códigos más largos. La sección de codificación es muy parecida a la de los primeros programas. Cada letra del texto es leída y convertida en un número entre 1 y 26. Al número equi-



valente se le aplilca un factor de escala de cinco y la subcadena de puntos y rayas que resulta es lo que se imprime (línea 100).

Parà la subrutina de descifrado que constituye la parte final del programa, se utiliza una técnica de búsqueda (línea 225).

Después de leer las señales Morse, el ordenador buscará en la serie hasta que encuentre una subserie idéntica. Cuando esto suceda, es inmediato transformar la posición de la serie en el correspondiente código númerico ASCII. Nuevamente se utiliza la función CHR$ para sacar el resultado requerido.

```
10 CLS:DIM A$(26):F$="      "
   :KEY OFF
15 POKE -853,255
20 FOR X=1 TO 26:READ A$(X)
   :NEXT X
30 INPUT"Codificar (1) o
   decodificar (2) ";R
40 IF R=2 THEN 140
60 INPUT"MENSAJE A
   CODIFICAR";M$
70 FOR X=1 TO LEN (M$)
80 IF MID$(M$,X,1)=" "THEN
   PRINT"   ";:GOTO110
90 P$=MID$(M$,X,1)
100 PRINT"  "
   ;A$(ASC(P$)-64)
110 NEXT X
120 PRINT:PRINT:PRINT
   TAB(4);"PULSA UNA TECLA
   PARA COMENZAR";
125 IF INKEY$="" THEN 125
130 RUN
140 INPUT"MENSAJE A
   DECODIFICAR";M$:M$=M$
   +" "
160 FOR X=1 TO LEN(M$)
170 K$=MID$(M$,X,1)
180 IF K$=" " THEN 220
190 S$=S$+K$
200 NEXT X:GOTO120
210 IF LEN(S$)>5 OR LEN(S$)
   <1 THEN PRINT"ERROR"
   :GOTO 120
220 L=LEN(S$):IF MID$(S$,1
   ,L)!=" "THEN S$=MID$
   (S$,1,L-1)
225 FOR H=1 TO 26:IF S$=A$
   (H) THEN PRINT CHR$(H+64)
230 NEXT H
240 S$="":GOTO 200
250 DATA 0-,-000,-0-0,-00,0
   ,00-0,--0,0000,00,0---
   ,-0-,0-00,--,-0,---,0--
   ,--0-,0-0,000,-,00-,000-
   ,0--,-00-,-0--,--00
```

NO OLVIDES EL TELEFONO... 

Cuando, por cualquier motivo, nos escribas, no olvides indicar tu número de teléfono. Así nos será más fácil y rápido ponernos en contacto contigo.

Gracias.

CUIDADO DE CINTAS Y DISKETTES

- PROTEGER LA INFORMACION
- HACER COPIAS DE SEGURIDAD
- INDEXAR
- ENVIO POR CORREO CINTAS Y DISCOS

Incluso si tú eres una de esas personas que guardan su información importante en la parte posterior de un sobre ya utilizado, cuando guardes ésta en cinta o *diskette*, desde luego, no te merecerá la pena ser desorganizado.



Los sistemas de almacenamiento basados en cintas o discos magnéticos hacen posible que el usuario de ordenadores personales pueda guardar grandes cantidades de información, o miles de programas, de una manera extremadamente compacta.

Pero la eficiencia que proporcionan los sistemas de almacenamiento magnético es también un potencial punto débil. Debido precisamente a que una diminuta cinta o *diskette* pueden contener tanta información, cualquier daño que se les produzca se convierte en un resultado desastroso. Aquí, el soporte magnético es un material particularmente vulnerable.

SEGURIDAD

Obviamente, lo más importante es asegurarse de que una vez depositada la información en la cinta o *diskette* ésta permanecerá allí, pudiendo ser recuperada siempre que lo deseemos. Pero existen dos clases de daños que pueden afectar a este tipo de soporte: el físico y el magnético.

El daño físico puede ser cualquier cosa, desde doblar un *diskette* o triturar una cinta, hasta acumular suciedad en su superficie. Esto se previene fácilmente, protegiéndolos siempre que no se estén utilizando. Se deben guardar en su sobre e introducirlos en una caja adecuada para *diskettes*. Las cintas también deberán meterse dentro de sus cajas y, preferiblemente, emplear uno de esos módulos de plástico que hay para apilar un buen número de cajas. Finalmente, cabe recomendar su almacenamiento lejos del calor, la humedad y el polvo.

Tampoco se debe tocar la superficie magnética de un *diskette* o el trozo de cinta que asoma por un *cassette*. También es un buen hábito rebobinar las cintas hasta su comienzo, una vez que hemos acabado de utilizarlas; esto significa que dejamos al descubierto la parte menos vulnerable de la misma. Asimismo logramos que la cinta esté disponible para su inmediato uso en cualquier momento. No es conveniente dejar olvidados los *diskettes* y las cintas en sus correspondientes unidades durante largos períodos de tiempo después de utilizar el ordenador.

Existen determinados campos magnéticos que tienen fuerza suficiente para alterar el contenido de una cinta o *diskette*. Estos proceden de diversos electrodomésticos habituales en el hogar. Por ejemplo, los altavoces contienen potentes imanes, y algunos motores eléctricos también. Por lo general, cintas y *diskettes* se deben mantener bien alejados de cualquier equipo eléctrico, incluyéndose la televisión.

MAS SEGURIDAD

El accidente puede tener lugar, por lo que merece la pena hacer copias de seguridad o tomar algún tipo de precauciones extraordinarias, especialmente para las grabaciones más importantes.

Por lo general, es una buena idea grabar dos versiones de todo, incluso en la misma cinta o *diskette*. Existen menos probabilidades de que se deterioren dos que una sola. Pero si el fichero es realmente importante haz otra copia en distinto lugar. Se almacenará en otro sitio y preferiblemente no se utilizará. Otro consejo es guardar en cinta copias de los programas existentes en *diskette* —la cinta es menos delicada.

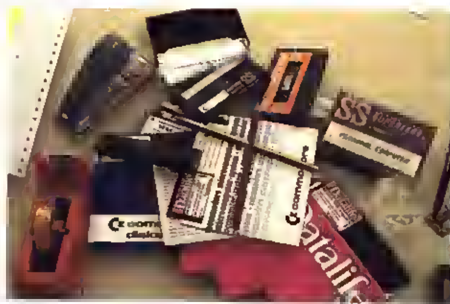
Proteje tus cintas contra la sobreescritura quitando las lengüetas

posteriores del *cassette*. Los *diskettes* pueden ser protegidos igualmente con sólo pegarles una etiqueta adhesiva en las muescas laterales de la funda.

SIGUE LA PISTA

Las cintas y *diskettes* pueden haberse utilizado hace mucho tiempo, por lo que resultará difícil recordar, con exactitud, lo que hemos depositado en ellos. Una regla general consiste en etiquetar las cosas tanto como podamos. Si tu economía te lo permite, no está de más almacenar cada fichero por separado, en cintas cortas, o reservando *diskettes* para ficheros relacionados entre sí.

Proporciona a cada fichero un nombre claro y único (dentro de los límites que permita tu sistema). En particular, si tienes varios desarrollos de un mismo programa, dale a cada uno un nuevo nombre (o número). Escribe los nombres de los ficheros



en la etiqueta del *cassette*, así como en la tarjeta que va en la caja, pues ambas deben ir asociadas en caso de separación. No escribas en las etiquetas de un *diskette* con un lápiz duro porque te puedes cargar el *diskette*. Es preferible emplear un rotulador de punta de fieltro. Un libro que actúe como índice de ficheros será de gran ayuda. En él puedes indicar todos los nombres, dónde está el fichero y cualquier otra anotación. Merece la pena incluir una sentencia tipo REM en el fichero, en la que se anota la fecha y descripción.

ENVIO DE FICHEROS

Las cintas y *diskettes* son una manera muy conveniente de enviar información o programas por correo. Las cintas son razonablemente fuertes y viajan bien. Sorprendentemente, lo delicado y más propenso a rotura es la cajita, por lo que si la evitas, además ahorrarás peso. Aunque difíciles de encontrar, existen carteritas para envío por correo, pero otro tipo de bolsas almohadilladas son perfectamente adecuadas.

El mayor riesgo que corren los *diskettes* es el de ser doblados. Por ello un bocadillo hecho con dos láminas fuertes evita este problema. Es muy importante disponer un mensaje en parte visible del paquete, en el que se lea algo así como: MATERIAL MAGNETICO, MANEJESE CON CUIDADO. Para mayor seguridad, el correo certificado suele ser una buena solución.

LOS MEJORES DE INPUT

Hemos pensado que es interesante disponer de un ***ranking*** que ponga en claro, mes a mes, cuáles son los programas preferidos de nuestros lectores. Para ello, es obligado preguntaros directamente y tener así el mejor termómetro para conocer vuestras preferencias. Podéis votar por cualquier programa aunque no haya sido comentado todavía en **INPUT**.

El resultado de las votaciones será publicado en cada número de **INPUT**.

Entre los votantes sortearemos 10 cintas de los títulos que pidáis en vuestros cupones.

Nota: No es preciso que cortéis la revista, una copia hecha a máquina o una simple fotocopia sirven.

Enviad vuestros votos a: **LOS MEJORES DE INPUT** Alberto Alcocer, 46 - 4.° B. 28016 Madrid

ELIGE TUS PROGRAMAS

Primer título elegido ____________________

Segundo título elegido ____________________

Tercer título elegido ____________________

Programa que te gustaría conseguir ____________________

Qué ordenador tienes ____________________

Nombre ____________________

1.er Apellido ____________________

2.° Apellido ____________________

Fecha de nacimiento ____________________

Teléfono ____________________

Dirección ____________________

Localidad ____________________

Provincia ____________________

INPUT MSX N.° 1

MSX; HACIA LA COMPATIBILIDAD

- LA NORMA MSX
- DESCRIPCION DE LOS CHIPS
- INTERFACES OE PERIFERICOS
- COMPATIBILIDAD ENTRE OISTINTDS MOOELOS

Indudablemente, el tema de la compatibilidad presenta unos problemas que traen de cabeza a multitud de personas. Por un lado, los que trabajan en las áreas de *software* y de desarrollo de periféricos y por otra parte, el sufrido consumidor.

El usuario al que va destinado el ordenador personal, se sorprenderá ante las innumerables opciones que el mercado ofrece. Aparecen nuevos aparatos, periféricos, programas y un sin fin de elementos adicionales, que más que simplificar la elección de un ordenador, la complican demasiado.

Actualmente estamos viviendo un *boom* (ahora no tan espectacular como hace varios años, ante la aparición del **ZX Spectrum** y los **Commodore**) en la industria del ordenador doméstico. Tenemos multitud de opciones, entre las que podremos encontrar aparatos que tienen desde 16K de RAM hasta 64K, siendo este, en la mayoría de los casos, junto con el precio de la máquina el baremo que se utiliza en la compra de un ordenador.

También se suele tener en cuenta la disponibilidad de *software* que existe para ese ordenador en particular, puesto que a nadie le agradaría comprarse un aparato si luego apenas hay programas para soportarlo. Ante todos estos problemas, han llegado los japoneses intentando poner punto final a semejante *maremagnum* de ordenadores, periféricos y, en suma, acabar con el tan temido tema de la incompatibilidad de aparatos.

Efectivamente, mientras que en el resto del mundo, los americanos con el **Commodore 64** y el **Vic 20** y los ingleses con el **ZX Spectrum**, y ahora, el **Amstrad**, se repartían el mercado, los japoneses se han mantenido a la espectativa sin competir con estos dos potentes paises. Sólo se limitaron a lanzar pequeños ordenadores tales como el **Sord M5** y el **Sharp MZ700**. Mientras tanto, en el archipiélago nipón, había nacido una norma que poco a poco se iría extendiendo a todos los paises occidentales. Su fuerza radica en algo que ningún fabricante había conseguido hasta el momento; la compatibilidad entre los ordenadores.

La historia del desarrollo de este estándar comenzó en el año 1983, cuando **ASCII/Microsoft** y el ingeniero **Kay Nishi** llegaron a un acuerdo para comercializar un BASIC particular para más de una docena de casas que deseaban incorporar entre sus productos, un ordenador personal. Este lenguaje se denominó **BASIC MSX** y fué el inicio de la estandarización, ya que esta no sólo se limitó a que todos los aparatos tuviesen el mismo lenguaje, sino que reuniesen una serie de características comunes que permitieran la compatibilidad de los productos que se fabricasen bajo el logotipo de **MSX**. De esta forma, el 17 de Junio de 1983 se presentó oficialmente en Tokio, el ordenador **MSX** tal y como lo conocemos ahora, iniciando así un nuevo concepto del ordenador doméstico.

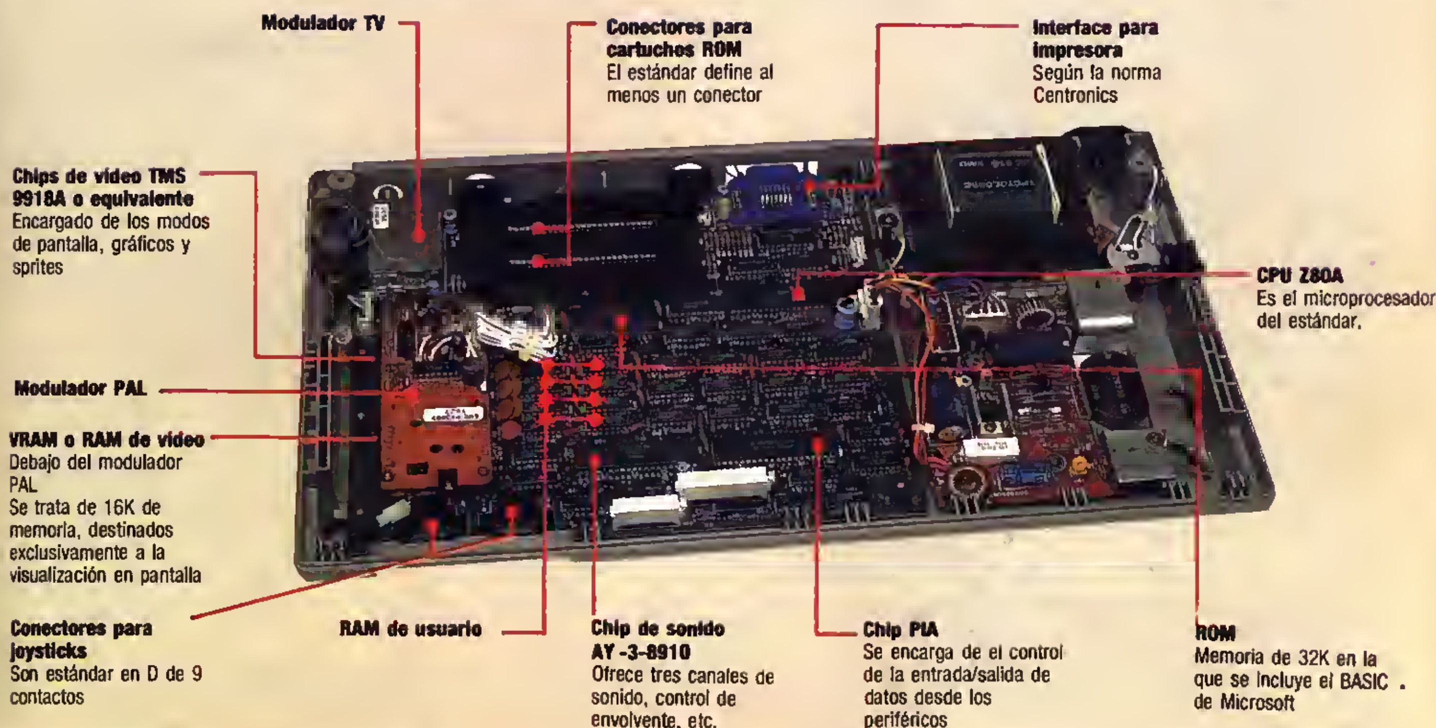
CARACTERISTICAS DEL MSX

Estos ordenadores reunen una serie de elementos, que como ya hemos dicho anteriormente, son comunes a todos ellos. Las características de los ordenadores **MSX** son:

- un microprocesador Z-8DA fabricado por **Zilog** o equivalente
- un *chip* de video tipo TMS 9918A de **Texas Instruments** o equivalente
- un *chip* de sonido tipo AY-3-891D de **General Instruments**
- 32K de memoria ROM, donde va incorporado el **BASIC MSX** (MicroSoft eXtended BASIC).







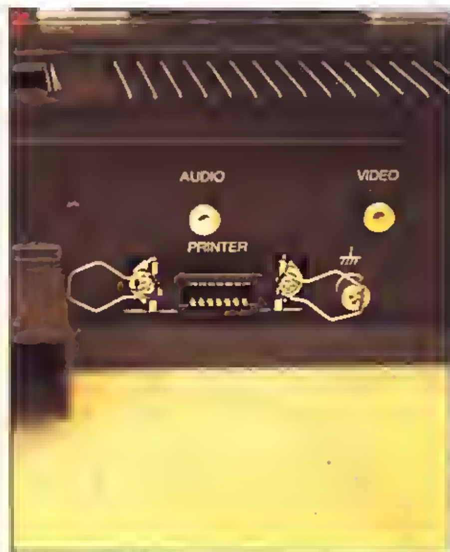
- al menos 8K de RAM (aunque es recomendable que sea superior a 16K)
- *display* de 40 columnas
- capacidad para generar 16 colores
- conector a *cassette* con dos velocidades posibles
- bus de expansión para cartuchos o *interfaces*
- diversas opciones de teclado para los mercados de Japón, Corea, Europa y Estados Unidos, y
- un *port* para *joysticks*.

Como podemos observar, un ordenador con estas especificaciones es, de por sí, bastante completo y todas, absolutamente todas las máquinas que se comercialicen bajo el logotipo **MSX** han de cumplir estas estrictas normas. Sólo así, el usuario podrá tener la certeza de que cualquier periférico, programa o elemento adicional que adquiera para un **MSX** funcionará correctamente en un ordenador **MSX**.

Ahora bien, estas son las características mínimas, puesto que se deja total libertad al fabricante para añadir o alterar (sin omitir característica alguna de las expuestas anteriormente) cualquiera de estos elementos. En el mercado encontraremos múltiples opciones. Cabe destacar, por ejemplo, que la mayoría de los fabricantes incorporan dos *ports* para *joysticks* (recordemos que en las especificaciones básicas, se indica que ha de haber al menos uno), mientras que otros incorporan al ordenador *software* específico, tal como bases de datos, agendas, tratamiento de textos, etc. En suma, podrás conectar a un ordenador **Sony**, una unidad de *diskettes* de **Philips**, una impresora **Toshiba**, etc., sin necesidad de *interfaces* extraños (caso del **Commodore**), ni *software* específico (caso del **Spectrum**).

Pero estudiemos los componentes más de cerca y veamos sus posibilidades.





DESCRIPCION DE LOS COMPONENTES

Aunque sea repetitivo, los componentes que veremos a continuación están en todos los ordenadores **MSX**. Veamos cuales son y la función que desempeñan:

- UNIDAD CENTRAL DE PROCESO (UCP). Este *chip* es el ya conocido y de una fiabilidad más que probada, Z-80A de **Zilog**, que funciona a una frecuencia de reloj de 3.57954 Mhz. Es un *chip* tachado de antiguo, ya que tiene más de cinco años, pero que parece gustar a los fabricantes de ordenadores. No obstante es el más utilizado ya que lo incorporan ordenadores tales como el **Spectrum**, **Amstrad**, etc. La UCP es el corazón de todo ordenador, ya sea grande o pequeño. Es el que ordena y el que hace cumplir las instrucciones que el usuario introduce mediante un programa.

- MEMORIA. Está compuesta de una ROM de 32K con el BASIC de **Microsoft** incorporado y una RAM de al menos 8K, teniendo en cuenta que ambas son ampliables. La memoria ROM es fija y en todos los casos sólo se limita a ejecutar el programa residente en ella, en este caso el BASIC. Sin embargo, la memoria RAM, puede contener cualquier tipo de información y ha de ser al menos 8K, aunque en Europa la configuración mínima tiene 16K.

- PROCESADOR DE VIDEO. Una característica única en estos aparatos es la de poseer 16K de VRAM (memoria para video). Esto se debe al *chip* 9918A de **Texas Instruments** o equivalente, que controla todo lo que aparece en la pantalla. Permite utilizar hasta 16 colores y definir hasta 32 *sprites*, pudiendo situar 4 en una misma línea. Posee además su propio juego de instrucciones, lo que le confiere una sofisticación más allá de lo usual.

Este *chip* facilita 4 modos de resolución en la pantalla. Dos modos de texto y dos modos gráficos. En los modos de texto tenemos la posibilidad de visualizar 40 columnas × 24 filas ó 32 columnas × 24 filas, mientras que en los modos de gráficos existe la posibilidad de representar 256 × 192 *pixels* ó 64 × 48 *pixels*. La forma en que se eligen el tamaño de los *pixels* es directa, es decir, basta con indicarlo dentro de la instrucción SCREEN, pero ya la veremos en su momento.

- EL GENERADOR DE SONIDO. Se trata del *chip* de **General Instruments**, AY-3-8910 o equivalente, capaz de generar tres tonos y ocho octavas. Posee 10 sonidos pre-programados y sus posibilidades son multiples, gracias a la combinación de diversas tonalidades y los distintos nivles de ruido que se pueden conseguir.

- INTERFACE DEL CASSETTE. Es un modulador FSK con dos posibilidades con respecto a la velocidad de transferencia, 1200 y 2400 baudios, según lo indiquemos directamente en las instrucciones CLOAD o SCREEN. Se puede conectar cualquier *cassette* normal al ordenador, procurando evitar en lo posible, utilizar *cassette* estéreo.

- BUS DE EXPANSION. Es un conector de 50 terminales que permite la conexión de cartuchos ROM o cualquier *interface* específico, como por ejemplo, el lápiz óptico, unidad de *diskettes*, etc.

- JOYSTICK. Es del tipo D con nueve patillas. Todos los ordenadores poseen al menos uno, pudiendo utilizarlos con casi todos los programas del mercado.

- INTERFACE DE IMPRESORA. Aunque existe la posibilidad de poseer un *interface* RS-232C, todos los **MSX** vienen con un *interface* paralelo **Centronics**. El conector permite utilizar cualquier tipo de impresora o periférico que tenga este tipo de *interface*.

- INTERFACE PROGRAMABLE PARA PERIFERICOS. Pertenece a **Intel** y es de tipo I-8255 o equivalente. Su misión es muy importante, ya que controla la entrada y salida de datos en *cassette*, *diskette*, etc. En realidad es el que permite que el ordenador pueda conectar con el mundo exterior.

Estos son los componentes que hacen posible el **MSX**. Como ya indicamos anteriormente, son las características básicas, pues los fabricantes tienen total libertad a la hora de diseñar sus aparatos como más gusten. En el mercado, encontramos varias adaptaciones y todas ellas muy interesantes. Podemos encontrar desde una base de datos que incorpora el **Sony HB-75P** hasta la unidad de superposición de video que tiene el **Pioneer PX-7**, pasando por un auténtico sintetizador musical como lo es el **Yamaha CX5M.**

Es importante destacar un punto interesante de los MSX, son ordenadores personales que permiten desarrollar las funciones más diversas y variopintas de los que hay en el mercado sin necesidad de *interfaces* o dispositivos adicionales. Su capacidad de generar gráficos y sonido van más allá de lo que cualquier ordenador doméstico tiene, y su potente BASIC, aún siendo el mejor de todos los lenguajes de los ordenadores de esta categoría, es muy sencillo de aprender y manejar, lo que facilita en gran medida su aprendizaje y aplicación.


ABU SIMBEL
PROFANATION
AZPIRI
MSX
Tiendas y distribuidores (91) 447 34 10
Pedidos contra reembolso (91) 715 00 67
DINAMIC

EL LADRON DE COCHES

El jeque **Abdul Al Kohol**, soberano del reino de **El Khazi**, tiene una afición en la que ha invertido gran parte de su fabulosa fortuna: los automóviles. Tiene cientos y cientos de ellos escondidos en las cámaras de su palacio, en salas de una complicada arquitectura, plagadas de escaleras, pasadizos, cintas transportadoras, etc. El jugador, **Mick**, desarrolla las funciones de mecánico de la flota de automóviles, encargándose de su cuidado y limpieza. Hasta ahora **Mick** estaba contento con su trabajo, pero últimamente ha tenido problemas con el jeque, que a decidido pagarle su trabajo con ojos de oveja, material de escaso valor para **Mick**. Por ello no es de extrañar que nuestro protagonista se haya enfadado y haya decidido robarle al jeque sus preciosos automóviles. Con



esta intención, **Mick** tendrá que desplazarse por todas las salas de palacio, llenando los coches del petróleo que gotea desde un depósito situado en el techo de cada sala.

Llegar hasta el depósito no va a ser fácil. Habrá que subir y bajar escaleras, saltar al estilo **Tarzán** ayudándose de lianas, y otras locuras por el estilo. Pero no es esto lo peor. Resulta que el jeque se ha enterado de la maniobra de **Mick** y ha mandado contra él a toda su guardia, compuesta por feroces espadachines, que no dudarán en acabar con una de las vidas de **Mick**, cada vez que se



encuentren con él. **Mick** sólo puede luchar contra ellos dejando caer gotas de petróleo por el suelo. Cuando los guardias pisen estas gotas, resbalarán y desaparecerán al caer. Las salas del palacio, ventiseís en total, están construidas a base de imaginación. Todas ellas incluyen elementos que las diferencian de las pantallas precedentes.

Mick y sus perseguidores son divertidos *sprites*, que se mueven con soltura, aunque sin demasiada naturalidad, por los pasillos y salas de palacio. El grado de dificultad del juego es suficiente como para mantener el interés del jugador

DATOS GENERALES

TITULO
Turmoil

FABRICANTE
Bug Byte

CLASE DE PROGRAMA
Juego

FORMATO
Cassette

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	8
INTERES	7
GRAFICOS	8
COLOR	7
SONIDO	6
TOTAL	**36**



durante bastante tiempo, sin que llegue el cansancio. Sobre todo se agradece la complejidad de algunas de las pantallas, que obligarán a pensar cómo llegar hasta el petróleo.

****

OFENSIVA INTERGALACTICA

Misión Espacial se enmarca dentro del más clásico estilo de los programas de marcianitos, aunque tiene una cierta mezcla de elementos procedentes de otros conocidos juegos. Narra la historia de un ataque de la **Confederación Intergaláctica** contra la Tierra, del que sólo el intrépido jugador, al mando de su *joystick*, será capaz de salvarnos. Con este objetivo, el jugador se enfrenta a la misión de activar la defensa de misiles, situada en un imaginario planeta próximo a nuestra querida Tierra.

Esta misión se desarrolla en dos fases diferentes, cada una en un escenario distinto En la primera fase, se trata de llevar, desde una base en tierra hasta la nave nodriza, cuatro misiles defensivos. El transporte se realizará mediante una pequeña nave que tendrá que llegar al suelo desde la nave nodriza. Las

fuerzas gravitatorias del planeta actúan sobre la nave de transporte, tirando de ella hacia el suelo. Para evitar un choque y la consiguiente destrucción de la nave de transporte, el jugador debe actuar sobre los cohetes propulsores, verticales y horizontales, tratando de conducir la nave con mucha suavidad.
La inercia será otro gran obstáculo. Cuando la nave se está moviendo en una dirección, solo será posible detenerla si se actúa durante un cierto tiempo sobre los cohetes.
En la segunda fase, en otra pantalla distinta, hay que hacer el proceso inverso, llevando los misiles desde la nave nodriza hasta su plataforma de lanzamiento.
Para llevar a cabo su misión el jugador dispone de tres naves y de una cantidad limitada de combustible. Cada vez que este se agote, la nave de transporte caerá al suelo rompiéndose en mil pedazos.
El argumento, como podréis apreciar es clásico. Viene aderezado, en el apartado de realización, con unos gráficos bastante sencillos, algo de sonido y una cierta dosis de acción.

DATOS GENERALES

TITULO
Misión Espacial

FABRICANTE
Advance

CLASE DE PROGRAMA
Juego

FORMATO
Cassette

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	6
INTERES	6
GRAFICOS	6
COLOR	5
SONIDO	6
TOTAL	**29**





LAS CAVERNAS PERDIDAS

En las profundidades de una caverna subterránea, **Pitfall Harry**, nuestro héroe, lleva a cabo una desesperada búsqueda. Su esperanza es encontrar a su querida **Rhonda**, a su gato **Quickclaw** y el valiosísimo diamante **Raj**. Por fortuna, **Harry** es un hombre joven y ágil, capaz de correr, saltar, subir y bajar por peligrosas escalas de cuerda, nadar, etc.



En el interior de las cavernas, **Harry** va a tener que desarrollar todas estas habilidades.
En efecto, se trata de verdaderos laberintos, con pasillos y salas a diferentes niveles de profundidad, con simas y precipicios que habrá que saltar, con escalas de cuerda como único medio para llegar a determinadas zonas, con lagos, charcas y cascadas que **Harry** tendrá que cruzar a nado... Por si fuera poco para nuestro héroe tener que realizar todo tipo de proezas gimnásticas para moverse por las cavernas, además tendrá que enfrentarse con unos cuantos murciélagos, escorpiones, anguilas y otros seres dañinos, que intentarán evitar el éxito de su misión.
La caverna está dividida en varios tramos por una serie de cruces dibujadas en el suelo. Cada vez que **Harry** llegue a una de las cruces, habrá completado un tramo. Si en el tramo en el que se encuentra, tiene la mala fortuna de topar con alguna de las alimañas mencionadas, tendrá que retroceder en la caverna sólo hasta el principio del tramo, es decir, sólo hasta la última cruz que hayan pisado sus pies. Este procedimiento sustituye al más clásico de un número de vidas limitado y le da un cierto atractivo al juego, empujando al jugador a buscar la siguiente cruz y a completar así un nuevo tramo.
Es realmente entretenido explorar la caverna, descubriendo cada vez nuevos paisajes y escenarios.
Técnicamente el juego es muy bueno. Los programadores contaban con la ventaja de trabajar con un juego que ya fue un éxito en su dia en otras máquinas como el **Commodore 64**. Sólo tenían que preocuparse de la adaptación a **MSX**.
Aspectos como el *scrolling* que se produce, de una forma continua y prefectamente suave, al avanzar por la caverna, o efectos como el de movimiento de las olas, cuando **Harry** está nadando en alguno de los lagos subterráneos, seguramente harán las delicias de cualquier jugador.
Además el juego cuenta con un elemento esencial que lo convierte en muy adictivo: es la emoción que se experimenta al ir descubriendo los diferentes paisajes y escenarios de la caverna.
Todas estas características hacen de **Pitfall II** un juego de total vigencia, teniendo en cuenta que se trata de la adaptación a MSX de una versión, que cuenta aproximadamente con dos años de antiguedad.

DATOS GENERALES

TITULO
Pitfall II

FABRICANTE
Activision

CLASE DE PROGRAMA
Juego

FORMATO
Cassette

CALIFICACION (Sobre 10 ptos.)	
ORIGINALIDAD	8
INTERES	8
GRAFICOS	7
COLOR	8
SONIDO	6
TOTAL	37





BUSCANDO LA SALA DEL MAGO

Cuarenta dias y cuarenta noches es el plazo que le ha sido concedido al explorador **Sabreman** para tratar de librarse del hechizo que le tiene preso y que, noche tras noche, le convierte en un triste e irracional hombre-lobo. El hechizo se cumplirá por completo cuando transcurra este plazo, la transformación será permanente y el pobre desdichado terminará sus días no como hombre, sino como lobo. Ante tan triste perspectiva no cabe perder el tiempo. Hay que hacer algo y deprisa. Quizá para empezar convenga examinar la situación en que se encuentra nuestro hombre. Si observamos atentamente descubriremos que se encuentra en una sala de pétreos muros, en la que hay varias salidas. Esta sala es una de las muchas de que consta el castillo del mago, una complicada construcción de piedra, que ha contemplado el paso de muchos y muchos siglos y entre cuyos muros se esconden cierto número de objetos, custodiados por extrañas criaturas y artificios y mecanismos más extraños aún. En una de estas salas hay un caldero puesto al fuego en el que hierve una extraña pócima, y junto a este caldero está **el mago**, dueño y señor del castillo. El es la única persona en el mundo que conoce el secreto para librar a **Sabreman** de su triste destino. Este secreto no es más que una lista de objetos, con un orden determinado, que aquel que quiera librarse del hechizo, tendrá que arrojar en el caldero en el que hierve la pócima. Los objetos, que se encuentran desperdigados aleatoriamente por las salas del castillo, son en total 32, pero sólo ocho de ellos son distintos, por lo que para conseguir los 32 se ha repetido cada objeto cuatro veces. Nuestro desdichado héroe, **Sabreman**, habrá de buscar y recoger, a través de las salas del castillo, todos y cada uno de los objetos que el mago le vaya indicando. En algunas de las salas el objeto estará al alcance de la mano sin ninguna dificultad, en otras sin embargo habrá que discurrir, cavilar, saltar y correr a gran velocidad hasta encontrar la forma de llegar al mismo.

DATOS GENERALES

TITULO
Knigth Lore

FABRICANTE
Ultimate

CLASE DE PROGRAMA
Juego de Aventura

FORMATO
Cassette

CALIFICACION (Sobre 10 ptos.)

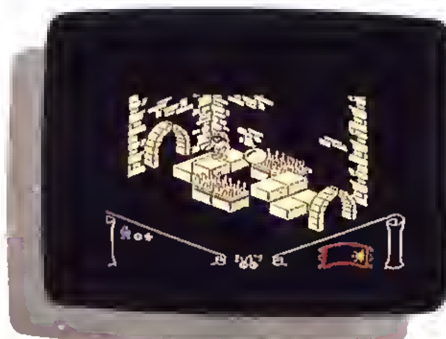
ORIGINALIDAD	9
INTERES	9
GRAFICOS	8
COLOR	6
SONIDO	7
TOTAL	**39**

Por si fueran pocas las dificultades que se le plantean al hombre-lobo, añadiremos una más. Consiste esta en no poder llevar más de tres objetos al mismo tiempo. De este modo, lo mismo que un objeto puede ser cogido en una sala, habrá de ser dejado temporalmente en otra. Al cabo de un rato es probable que haya que recoger el objeto que se dejó, lo que obligará a recordar en cual de las salas ha quedado. Probablemente, sólo aquel que haya ido anotando en un plano el camino recorrido, será capaz de salir airoso. Estos son algunos de los ingredientes de una de las mejores aventuras gráficas diseñadas para un ordenador. Hay intriga y emoción, pero sobre todo hay originalidad y sorpresa en todas y cada una de las salas que el jugador irá descubriendo. Es un juego para dedicarle tiempo, que hará pensar al jugador y que le obligará a emplear a fondo todas sus habilidades para resolver los complicados problemas, sin aparente solución, que alguna sala plantea. No es un juego de velocidad, pero habrá que ser bastante rápido para, en el transcurso de los cuarenta días y las cuarenta noches, reunir los ingredientes con los que preparar la pócima.

En fin, no queremos decir más. En estos juegos de misterio hay ciertos secretos que no conviene aclarar, al menos por el momento.







RESCATE EN LAS MINAS

DATOS GENERALES

TITULO
H.E.R.O.
FABRICANTE
Activision
CLASE DE PROGRAMA
Juego
FORMATO
Cassette

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	8
INTERES	8
GRAFICOS	8
COLOR	8
SONIDO	7
TOTAL	39

De la firma norteamericana **Activision**, reputada como creadora de algunos de los mejores juegos de ordenador, es original este **H.E.R.O.** La primera versión que tuvimos ocasión de probar, hace ya cerca de dos años, era para un ordenador de la marca **Commodore**. Ahora nos encontramos con el mismo juego, en versión **MSX**, y aunque dos años son muchos años en el vertiginoso



mundo de los programas de ordenador, no deja de sorprendernos la calidad de este juego y el que resulte totalmente vigente a estas alturas de 1986. No cabe duda de que hay juegos buenos que nunca pasarán de moda y que aunque deje de hablarse de ellos y no sean actualidad, siempre conservarán un atractivo especial, un cierto encanto que los convertirá en imprescindibles de toda buena biblioteca de programas. Este es exactamente el caso de **H.E.R.O.**

La trama se desarrolla en el interior de una profunda mina, en la que ha quedado atrapado un minero. Sólo hay un hombre capaz de salvarle y es el especialista **Roderick Hero**, aventurero y explorador de minas y cuevas subterráneas. **Roderick** cuenta con un buen equipo de salvamento, constituido por una mochila de transporte aéreo, una pistola laser y un paquete de cargas explosivas que podrá depositar en el suelo y hacer estallar cuando lo desee.

Mediante su mochila podrá



desplazarse arriba y abajo a través de los pozos y galerías de la mina. También podrá desplazarse horizontalmente, pero para ello habrá de ser muy preciso en sus movimientos, para contrarrestar el efecto de la gravedad. Lo mejor para estos movimientos horizontales será, siempre que sea posible, que camine sobre el suelo de la galería. Y decimos siempre que sea posible porque en determinados tramos el suelo estará constituido por hirviente lava volcánica. En este caso es mejor mantenerse en el aire.

La pistola laser será de utilidad para deshacerse de las arañas, murciélagos y demás bichos que habitan el interior de la mina, custodiando la entrada a algunos de los pozos y galerías. Un par de disparos laser y el camino quedará libre.

Por su parte, las cargas explosivas serán la única forma de seguir adelante en algunos tramos. Servirán para volar muros que impidan el acceso a una galería. **Roderick** tendrá que acercarse todo lo que pueda al



muro, soltará entonces la dinamita y se alejará, rapidamente, todo lo que pueda para ponerse a salvo antes de que se produzca la explosión.

La dinamita, lo mismo que el número de vidas de nuestro héroe, es un bien limitado que no habrá que derrochar.

Esta trama se desarrolla en unos escenarios gráficos excelentes, llenos de color y de detalles divertidos, como por ejemplo los farolillos de gas que aparecen en determinadas galerías y que, si se apagan, dejarán a **Roderick** buscando la salida en la más completa obscuridad.

También es excelente el movimiento de los escenarios (*scrolling*) que resulta sumamente suave y, sobre todo, el del protagonista, que se desplaza respondiendo perfectamente a la más leve presión sobre el *joystick*.

El juego se desarrolla a través de una serie de niveles de dificultad creciente. Cuando se rescata al minero se accede al siguiente nivel, en el que la misión resultará más complicada.

CUIDADO CON LAS CURVAS

Entre los juegos de ordenador que más han llamado la atención al autor de este comentario, siempre han ocupado un lugar muy especial los dedicados a simular carreras de coches. Desde aquellos primeros tiempos, cuando hizo su aparición el célebre **Poole Position** de **Atari**, hasta nuestros días este tipo de juegos ha experimentado una importante evolución, que ha afectado sobre todo a la realización técnica más que a la filosofía general del juego que sigue siendo esencialmente la misma: la visión de la carretera, tal y como la vería el piloto de un automovil real, discurriendo velozmente ante nuestros ojos, en una sucesión constante de curvas y rectas. Tras esta evolución técnica se ha llegado a juegos de tan excelentes características como este **Hyper Rally**.



Para empezar, se trata de una larga carrera que se desarrolla en trece etapas, cada una en un circuito distinto y con un escenario de fondo diferente, pero siempre lleno de colorido. Cada una de las etapas es eliminatoria, es decir, hay que conseguir la suficiente puntuación para superarla antes de poder pasar a la etapa siguiente. Para ello habrá que fijarse en un valor, que podemos

DATOS GENERALES

TITULO
Hyper Rally

FABRICANTE
Konami

CLASE DE PROGRAMA
Carrera de coches

FORMATO
Cartucho ROM

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	9
INTERES	9
GRAFICOS	8
COLOR	7
SONIDO	8
TOTAL	**41**

denominar penalización de la etapa, que aparecerá en la pantalla al comienzo de la misma. A medida que el piloto cometa errores, su penalización aumentará. Si por el



contrario su conducción es correcta, se reducirá dicha penalización. Al término de la etapa y si el piloto ha conseguido llegar a la meta antes de agotar el depósito de combustible, su penalización podrá ser inferior a la de la etapa, lo que le permitirá seguir adelante, o podrá ser superior, en cuyo caso el piloto no tendrá más remedio que volver a empezar.



Aparte de este valor de penalización, existe una puntuación general para el piloto en la que intervienen dos factores. El primero de ellos es la velocidad media que se haya conseguido en el circuito. Cuanto mayor sea, mejor será la puntuación. El segundo factor lo constituye el número de rivales sobrepasados durante la etapa, cada uno de los cuales proporcionará 250 puntos al jugador. El coche, que se puede controlar desde el teclado o mediante un *joystick*, dispone de acelerador, freno y dos velocidades, una corta y una larga. Su comportamiento, sobre todo en las curvas, es bastante suave respondiendo perfectamente, con rapidez pero sin brusquedad, a la acción de control del jugador.
Los circuitos de las distintas etapas son bastante entretenidos. Estan plagados de curvas que pueden tomarse a gran velocidad, y en las que lo único que debe preocupar al jugador es poder esquivar los numerosos vehículos rivales.

★★★★★★★★★★★★★★★★★★★★★★★★

TODOS AL FUTBOL

DATOS GENERALES

TITULO
Soccer
FABRICANTE
Konami
CLASE DE PROGRAMA
Juego de Fútbol
FORMATO
Cartucho ROM

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	9
INTERES	8
GRAFICOS	9
COLOR	8
SONIDO	7
TOTAL	41

Los que conozcaís un poco cual ha sido la evolución de los juegos de ordenador, comprendereís que juegos como este **Soccer** de **Konami** hubieran resultado impensables hace tan sólo uno o dos años.
El juego es una representación auténticamente realista de un partido de futbol en el que se enfrentan dos equipos de once jugadores. ¿Os imaginaís lo que significa controlar simultaneamente a los 22 jugadores? Antes de que comience el partido vaís a tener la oportunidad de hacer una selección entre varias opciones.



En primer lugar podreís escoger entre jugar contra el ordenador o echar un partidito con algún compañero. Si os decidís por el duelo hombre-máquina, habreís de escoger el nivel de juego de vuestro oponente electrónico, entre cinco posibles. Al principio olvidaros de los niveles que no sean el uno o el dos si no quereís perder por muchos a cero. Hecho esto podeís decidir los colores de los dos equipos. Hay varios colores predefinidos para las camisetas y, curiosamente, también para el pelo de los jugadores. Así, por ejemplo, podreís escoger para vosotros a los jugadores rubios de la camiseta azul, mientras que vuestro adversario se decide por los morenos de la camiseta roja. Por cierto, hay rumores de que pronto estará disponible una versión del juego con las camisetas de varios equipos españoles.
Antes de empezar el partido sólo queda decidir la duración del mismo y los nombres que llevará cada equipo. A partir de entonces dará comienzo el juego. Los jugadores saldrán de los vestuarios entre el clamor del público aficionado y ocuparán su lugar alrededor del círculo central. El juego se desarrollará como un auténtico partido. Los jugadores de cada

equipo serán capaces de correr con el balón, entrar, tirar a puerta y hasta lanzarse en plancha para arrebatar el balón contrario. Controlarás el balón con el jugador que más cerca se encuentre del mismo, pero podrás hacer que el control pase a otro jugador sin más que pulsar el botón de disparo de tu *joystick*. Tendrás que tener en cuenta hasta las faltas y los fueras de juego (lo único que no te puede ocurrir es que alguien del público te arroje un bote de coca-cola).
Cuando el contrario llegue hasta tu portería vas a tener que utilizar a tu portero. Podrás moverle hacia ambos lados de la portería y hacer que se lance a por el balón en el momento oportuno. Si al acabar el segundo tiempo el resultado es un empate, entrarás automáticamente en el lanzamiento de penaltis, una de las fases más curiosas del juego.
Hay que destacar algunos aspectos cómicos o divertidos del juego, como por ejemplo la expresión de júbilo del jugador que acaba de meter un gol, al que verás recorrer medio campo dando saltos de alegría. Al





mismo tiempo verás como al portero al que le han metido un gol, da golpes con la mano contra el suelo, en un evidente signo de desesperación.
Técnicamente el juego es casi perfecto. Gráficos, sonido y color ofrecen la inconfundible calidad **Konami**. Quizás sólo una pega, el movimiento de los jugadores es un tanto brusco, un poco a saltos.

En conjunto el juego es tremendamente entretenido. Te recomendamos que, cuando lo tengas, reunas a unos cuantos amigos y organices todo un campeonato.

★★★★★★★★★

DATOS GENERALES

TITULO
Hyper Sports 1

FABRICANTE
Konami

CLASE DE PROGRAMA
Deportes

FORMATO
Cartucho ROM

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	8
INTERES	9
GRAFICOS	9
COLOR	9
SONIDO	8
TOTAL	43

HAGAMOS DEPORTE







Puede que no te encuentres en muy buena forma, quizás has engordado un poco últimamente y hayas pensado, o te hayan recomendado, hacer un poco de deporte. La decisión es dura, va a costarte trabajo al principio, pero verás como al cabo de unos días de entrenamiento empiezas a encontrarte en forma, más ágil y más dispuesto a todo.

A lo mejor no es mala idea que, si no encuentras el momento de empezar a hacer deporte, pruebes con este juego de **Konami** para armarte de valor. Se trata de un campeonato deportivo en el que tendrás que competir en varias pruebas. Para empezar te verás subido en el trampolín olímpico preparado para saltar a la piscina.

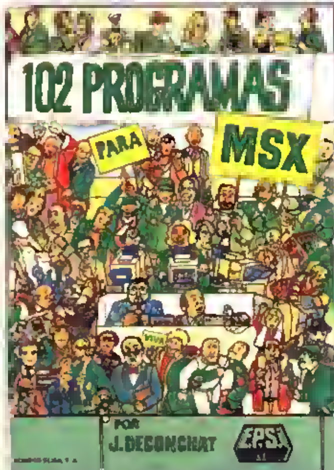
Pega un buen salto y luego comienza a girar en el aire. Cuantas más vueltas des antes de entrar en el agua, mayor será la puntuación que te otorguen los jueces. Vas a tener tres intentos en los que tendrás que obtener puntuación suficiente para clasificarte. Sólo así podrás pasar a disputar la siguiente prueba. Consiste ésta en el salto del caballo.

Tomando carrerilla tendrás que aproximarte al caballo a toda velocidad. Cuando estés sobre él, apoya las manos y sal disparado hacia arriba. Comienza entonces a girar tan deprisa como puedas hasta que llegues cerca del suelo. Pulsa entonces el botón de disparo para caer de pie. Si lo has hecho bien los jueces te permitirán pasar a la siguiente prueba.

En esta ocasión estarás de pie sobre una gran cama elástica. Debes empezar a saltar y a ganar altura (no hay ningún techo que te detenga). Luego tienes que dar vueltas en el aire una y otra vez antes de volver a la cama elástica. Si das vueltas suficientes dentro del tiempo asignado pasarás a la última prueba: la barra fija. Empezarás a dar vueltas agarrado a la barra hasta que decidas soltarte. Tendrás entonces que dar vueltas, esta vez en el aire, antes de caer al suelo, a ser posible de pie.

Estas son las cuatro pruebas en las que tendrás que competir. Puedes hacerlo tu solo o puedes organizar un campeonato con tus amigos. El estadio estará abarrotado de un público entusiasta que aplaudirá tu clasificación. El juego es muy atractivo en todos los aspectos, especialmente en el de los gráficos, divertidos aparte de bien realizados, y en el movimiento de los atletas que sólo puede calificarse de perfecto.





102 PROGRAMAS PARA MSX

Autor: J.Deconchat
Editor: Ediciones Elisa
Paginas: 239
Precio: ND

La oferta de este libro consiste nada menos que en 102 programas, recopilados por el autor y ordenados según su grado de dificultad, listos para ser tecleados en tu **MSX**:

Puede parecer que són demasiados programas, pero se trata de programas cortos, todos ellos en torno a las 40 o 50 líneas, por lo que no será necesario perder demasiadas horas tecleando.

El objetivo del autor no ha sido el de proporcionar listados completos, sino más bién el de ofrecer una serie de listados que sirvan de ejemplo a los distintos usos de las instrucciones BASIC.

Los programas estan ordenados, atendiendo a su grado de dificultad, en cinco niveles. En cada nivel se estudian algunas de las instrucciones BASIC más corrientes. Por ejemplo, en el primer nivel se estudian las sentencias LET, PRINT, GOTO, GOSUB... A medida que avanzamos por los niveles, pasamos a ver instrucciones mas específicas. Por ejemplo el segundo nivel se ocupa de las instrucciones de manejo de tablas y matrices, el tercero de las cadenas de caracteres, el cuarto de las instrucciones de lectura de teclado y las referentes a los gráficos de alta resolución. Así llegamos, paso a paso, y sirviéndonos de los numerosos ejemplos incluidos en los programas, hasta el quinto nivel, en el que se tratan aspectos específicos de los **MSX** como las colisiones entre *sprites* o la lectura y escritura en la memoria de pantalla.

Cada capítulo incluye en sus primeras páginas una explicación de las instrucciones BASIC que encontraremos en los programas. A continuación aparecen los listados, obtenidos a partir de una impresora matricial, precedido cada uno de ellos de un breve comentario en el que se explican las características del juego, la forma de jugarlo,la estructura del programa, y algo tan interesante como las posibles extensiones o modificaciones que podremos llevar a cabo con el fin de mejorar el programa.



MSX PROGRAMACION. GRAFICOS, COLORES Y MUSICA.

Autor: E. Lowy, A.E. Gallego, S. Mansilla
Editor: S.M.
Paginas: 398
Precio: 1.272

Una buena guía del lenguaje BASIC **MSX** es una herramienta imprescindible para todo usuario del estandar. Salvo excepciones, los manuales incluidos con el ordenador ofrecen una información condensada sobre el lenguaje, con pocos ejemplos sobre la utilización de las diversas instrucciones y comandos. De este modo, el usuario que desee aprender y familiarizarse con el lenguaje, pero que carezca de conocimientos previos, se encontrará un tanto desamparado. Por estas razones, debemos dar la bienvenida a títulos como este **MSX Programación**, que se publica con la intención de servir de guía para el aprendizaje del lenguaje BASIC MSX.

El libro es sobre todo didáctico. Esta estructurado en 20 capítulos y 4 apéndices, en los que se desarrollan dos aspectos del BASIC: las instrucciones de caracter general y las dedicadas a aspectos específicos como los gráficos, colores y música. Cada capítulo incluye una sección de ejercicios resueltos, en la que mediante programas simples se explica el uso de cada una de las instrucciones. Además, y al final del capítulo, se incluyen unos ejercicios de recapitulación, que servirán para que el lector practique lo que acaba de aprender. Los ejemplos son especialmente abundantes en las secciones de gráficos, colores y sonido.

En esta última sección, por ejemplo, los autores han incluido las partituras de varias conocidas canciones y a partir de ellas, nos enseñan como pasar de la partitura al programa BASIC correspondiente.

18 JUEGOS DINAMICOS PARA TU MSX

Autor: P. Monsaut
Editor: Noray
Páginas: 87
Precio: 690

Una de las fases por las que suele pasar todo aficionado a la informática es la de la pasión por teclear ávidamente todo listado que pueda ejecutar su ordenador. Es en momentos como estos en los que adquieren todo su sentido libros como el que presentamos.

18 Juegos dinámicos para tu MSX es una recopilación de listados de juegos, sencillos pero no por ello menos interesantes, en los que se hace uso de los gráficos, colores, sonido y animación propios de las máquinas **MSX**.

Cada juego incluye un pequeño texto introductorio en el que se explica, de forma breve y amena, el argumento del juego y la forma de jugar: estrategias, teclas de control de movimiento, teclas de disparo, etc.

Hubiera sido interesante que este texto incluyera una explicación de la estructura y las rutinas del programa. Al faltar esta explicación, el lector tendrá que hacer un pequeño esfuerzo personal si quiere hacer algo más que jugar con el programa.

A continuación viene el listado del programa . El procedimiento empleado ha sido el de imprimir directamente el listado obtenido de una impresora matricial. Como apenas se ha reducido el tamaño del original y como la impresión es correcta, hay suficiente contraste entre los caracteres, y los programas resultan perfectamente legibles. Los listados son cortos, con una medida de 100 líneas cada uno, por lo que teclear y corregir no ocupará demasiado tiempo. Así, el tiempo ahorrado se puede emplear en comprender cómo funciona el programa, cómo hace uso de las facilidades que proporciona el ordenador y cómo aplica diversas técnicas de programación.

Los programas son los clásicos de este tipo de libros: Bombardeo aéreo, Slalom, Serpiente, La pared... Juegos sencillos, principalmente de acción, que el lector podrá utilizar directamente o como soporte sobre el que construir programas más elaborados.



MSX

DYNADATA

NO LO DUDE

PRECIO INSUPERABLE

CONJUNTO:
ORDENADOR
MONITOR Y
CASSETTE
DYNADATA MSX

62.500



TARJETA
PROGRAMA.
EL NUEVO FORMATO DEL FUTURO.

MONITOR 12"
FOSFORO
VERDE

ADAPTADOR TARJETA

DATA CASSETTE

NUEVO

DYNADATA MSX
CON TECLADO EN
ESPAÑOL*

* Letras y signos iguales al teclado del PC de IBM.

PMR

Ordenador DYNADATA MSX con teclado en español	46.900
DYNADATA MSX con cassette y monitor de color	95.500
DYNADATA MSX y unidad de diskette de 5 ¼" de 360 Kbytes con monitor de fósforo verde	108.900
con monitor de color	141.000
TARJETA/PROGRAMA del juego LE MANS con adaptador, el cual sirve para cualquier tarjeta que usted adquiera. Precio especial con la compra del DYNADATA MSX	4.900

Con el DYNADATA MSX usted podrá:

- Divertirse con la amplia gama de juegos MSX.
- Aprender Informática y Basic con el curso autodidáctico y audiovisual.
- Llevar gestiones administrativas con los programas de proceso de textos, base de datos, contabilidad, stock, recibos, etc.
- Ayudar a sus hijos en sus estudios de 3.º a 8.º de EGB con los programas de Matemáticas, Lenguaje y Ciencias Naturales.
- Aprender idiomas tan necesarios como el inglés.
- Programar con los lenguajes: LOGO, PASCAL, FORTRAN y COBOL.

CURSO DE INFORMATICA Y BASIC

- Autodidáctico
- Audiovisual
- 12 cassettes
- 24 lecciones
- Evaluaciones periódicas
- Diploma Fin de Curso



Especificaciones DYNADATA MSX: Procesador Z80A, 64 Kbyte RAM, 16 Kbyte VRAM, 32 Kbyte ROM, 24 líneas x 40 columnas, 256 x 192 pixels, 16 colores, MSX-BASIC, MSX-DOS.

DYNADATA

Y:

SONY	PIONEER
PHILIPS	YAMAHA
CANON	MITSUBISHI
SANYO	GOLDSTAR
JVC	SAMSUNG
TOSHIBA	HITACHI
SPECTRAVIDEO	MATSUSHITA
	CASIO

Se han decidido por MSX. Esto le permite compartir los programas y periféricos con todas estas reconocidas marcas.

Por todo, NO LO DUDE. Decídase por

DYNADATA

Solicite información: Sor Angela de la Cruz, 24 - 28020 Madrid. Tels. (91) 279 21 85 - 279 28 01 - 270 01 93. Telex 44619 DYNA
Delegación Barcelona: Aribau, 61, entlo - 08011 Barcelona. Tels. (93) 254 73 04 - 254 73 03